HUMAN-CENTERED AI
THROUGH SCALABLE VISUAL DATA ANALYTICS

A Dissertation
Presented to
The Academic Faculty

By

Minsuk Brian Kahng

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Computer Science

Georgia Institute of Technology

December 2019

HUMAN-CENTERED AI
THROUGH SCALABLE VISUAL DATA ANALYTICS

Approved by:

Dr. Duen Horng (Polo) Chau, Advisor
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Shamkant B. Navathe
School of Computer Science
Georgia Institute of Technology

Dr. Alex Endert
School of Interactive Computing
Georgia Institute of Technology

Dr. Martin Wattenberg
Google Brain
Google

Dr. Fernanda B. Viégas
Google Brain
Google

Date Approved: October 8, 2019

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my amazing advisor Polo Chau. I truly enjoyed being his student. Polo has been a tremendous role model on how to be a great advisor and mentor. His cheerful attitude and encouragement has helped me stay positive, believe in myself, and successfully complete my PhD. I am forever grateful to him for his continued support, invaluable advice, willingness to help, and time devoted to me

I would also like to thank my thesis committee members. I thank Sham Navathe for his mentorship and support since the beginning of my PhD studies, Alex Endert for his inspiring and thought-provoking comments, Martin Wattenberg for his constructive feedback which I always find insightful, and Fernanda Viégas for her positive energy which makes me smile and feel confident.

I am very fortunate to receive advice and guidance from mentors and collaborators across diverse research areas. In particular, I thank John Stasko who fueled my interest in information visualization. I am indebted to collaborators from whom I have learned much, including Rahul Basole, Jilles Vreeken, and Jamie Morgenstern. I would also like to extend my gratitude to Rich Vuduc and Ed Chi for their advice on job hunting.

I had wonderful experience during my internships at the Google Big Picture team and Facebook Applied Machine Learning group. I especially thank Nikhil Thorat, Hal Abelson, Pierre Andrews, Aditya Kalro, Thomas Dudziak, and Hussein Mehanna.

I also have to thank all the fantastic members of Polo Club. Thanks to my academic siblings who spent the early years of Polo Club together: Acar Tamersoy, Robert Pienta, Shang-Tse Chen, Fred Hohman, and Nilaksh Das. I am indebted to many incredible undergraduate and master's students who I have worked with: Zhiyuan (Jerry) Lin, Dezhi

(Andy) Fang, Ángel (Alex) Cabrera, Will Epperson, Elias Khalil, Ganesh Parameswaran, Mayank Gupta, and Peter Polack.

Many thanks go to my friends and colleagues at Georgia Tech. I thank Georgia Tech VIS group members: Chad Stolper, Jaegul Choo, Arjun Srinivasan, Hyunwoo Park, and Bahador Saket. I am also grateful for my friends from Korea who helped me start my PhD life in a new place and had Korean lunch together at Klaus: Joonseok Lee, Seungyeon Kim, Changhyun Lee, Hyojong Kim, Hannah Kim, Jong Seok Park, and Yeongjin Jang.

I would also like to thank Carolyn Young, Arlene Washington-Capers, and Kristen Perez for their administrative support.

Finally, I dedicate this dissertation to my family. Thank you to my mom and dad for their unconditional love and support. And most of all, thank you to my wonderful wife Youjin Kong for her support and encouragement. I am so grateful to have you by my side. I love you!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

While artificial intelligence (AI) has led to major breakthroughs in many domains, understanding machine learning models remains a fundamental challenge. How can we make AI more accessible and interpretable, or more broadly, human-centered, so that people can easily understand and effectively use these complex models?

My dissertation addresses these fundamental and practical challenges in AI through a human-centered approach, by creating novel data visualization tools that are scalable, interactive, and easy to learn and to use. With such tools, users can better understand models by visually exploring how large input datasets affect models and their results. Specifically, my dissertation focuses on three interrelated parts:

(1) **Unified scalable interpretation**: developing scalable visual analytics tools that help engineers interpret industry-scale deep learning models at both instance- and subset-level (e.g., *ActiVis* deployed by Facebook);

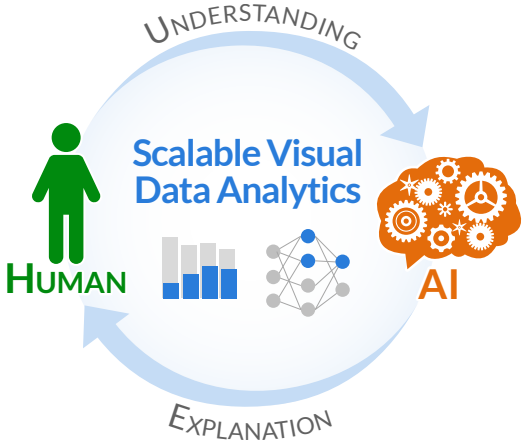
(2) **Data-driven model auditing**: designing visual data exploration tools that support discovery of insights through exploration of data groups over different analytics stages, such as model comparison (e.g., *MLCube*) and fairness auditing (e.g., *FairVis*); and

(3) **Learning complex models by experimentation**: building interactive tools that broaden people's access to learning complex deep learning models (e.g., *GAN Lab*) and browsing raw datasets (e.g., *ETable*).

My research has made significant impact to society and industry. The *ActiVis* system for interpreting deep learning models has been deployed on Facebook's machine learning platform. The *GAN Lab* tool for learning GANs has been open-sourced in collaboration with Google, with its demo used by more than 70,000 people from over 160 countries.

CHAPTER 1

INTRODUCTION



Machine learning (ML), or more broadly, **artificial intelligence (AI)**, has led to major breakthroughs in various domains. Recent success of *deep learning* has further accelerated this trend. Practitioners and researchers, including those without strong machine learning background, have been increasingly embracing these technologies. Companies and governments around the world are adopting machine learning for more of their products and services, such as image recognition, machine translation, conversational agents, recommender systems, medical diagnosis, and many more. These AI-powered systems are now transforming many aspects of our daily lives.

While powerful machine learning models have significantly improved the accuracy of many different tasks, people often have a difficult time understanding these models and interpreting the results produced from the models. Deep learning models are particularly difficult to understand, as complex mathematical functions and millions of parameters are used to train models. Because of that, people often use the models as "*black boxes*" without a deep understanding of why and how they work, which could be detrimental. For example, when the models do not perform satisfactorily, users would not understand the causes or know how to fix them. Even when the models work well, users may not trust them.

*How can we make artificial intelligence more **accessible and interpretable**, or more broadly, **human-centered***, so that people can understand their inner-workings more easily, build them more effectively, and use them more confidently? This dissertation presents ***new** paradigms, methods, and systems* that address this challenging problem.

1.1 Thesis Goal & Main Ideas

My dissertation addresses the fundamental and practical challenges in our understanding of machine learning models through a ***human-centered*** approach. Through my research in *visualization* and *data analytics* over the last decade, I realized that the key to solving this problem is bringing the *human* into the analytics process. By developing new ways for human to engage in the sensemaking process of machine learning models, we can promote people's understanding of complex ML systems. The goal of the dissertation is to ***build an interface from human to complex, large-scale machine learning systems**, where the interface is scalable, interactive, and usable*. With such interfaces, models that perform well will be understood and trusted, and those that do not can be interpreted and improved.

My key idea to tackling this problem is creating *novel **interactive data visualization** tools* that connect users with the machine learning models. The fields of *information visualization* and *visual analytics* have long been developing powerful interactive tools to help people explore and analyze data by amplifying human cognition. My idea is to build on this rich body of knowledge to design and develop novel interactive, visual tools that allow people to easily explore machine learning models to make sense of their underlying mechanisms, and effectively analyze their results for input datasets. Through these tools, I create new ways for AI to explain its reasoning, and for human to actively engage in the sensemaking process of machine learning models.

This research requires new visualization design principles, new data exploration models, and new scalable systems. This thesis presents solutions to these pressing needs.

1.2 Thesis Overview

Building interactive data visualization tools for the understanding of machine learning models involves many challenges. My dissertation is organized into three parts, each addressing one main research question and presenting example tools and methods that illustrate my answers, which are summarized in Table 1.1. The publications from this research are listed in Table 1.2. Next, I describe the main ideas behind the solutions.

Table 1.1: Three main research questions of thesis.

Research Question	**Answer**	**Example**
How to interpret large-scale models?	I. Unified Scalable Interpretation	Ch. 3
How to discover insights in workflow?	II. Data-driven Model Auditing	Ch. 4, 5
How to broaden access to complex models?	III. Learning by Experimentation	Ch. 6, 7

1.2.1 Part I. Unified Scalable Interpretation

Despite the recent interest in visualization for interpreting deep learning models, the large-scale datasets and the wide variety of models used in industry pose unique design challenges. While a common approach to interpreting machine learning models, called *instance-level analysis*, helps users explore a model's response for an individual instance, it does not easily scale. *How can we enable users to visually explore industry-scale modern deep models that use large and heterogeneous datasets?*

ActiVis*: Visual Exploration of Industry-Scale Deep Learning Models** (Chapter 3). We designed and developed *ActiVis* [86], a visual analytics system for interpreting large-scale deep learning models and results, through participatory design sessions with researchers and engineers across multiple teams at Facebook. With *ActiVis* (Figure 1.1), users can start their exploration with an overview of model architecture, and subsequently dive into localized inspection of activations. To support effective inspection of activations for large data, *ActiVis*'s visualization ***unifies *two commonly-used analytics patterns,* ***instance-***

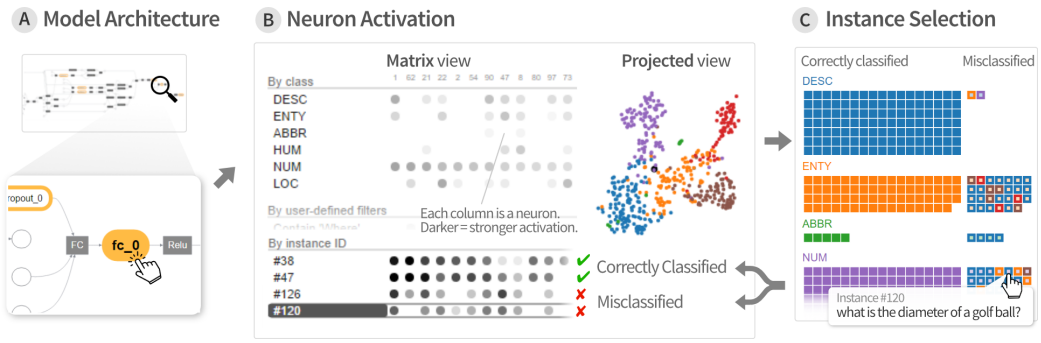


Figure 1.1: *ActiVis*'s multiple coordinated views help engineers explore complex deep neural network models at both instance- and subset-level.

level *and* ***subset-level analysis****, accelerating comparison of multiple instances and groups of instances for large datasets*, while most existing systems only support instance-level analysis. *ActiVis* has been deployed on Facebook's *FBLearner Flow* system used by most machine learning engineers at Facebook.

Table 1.2: Publications from each part of the thesis.

Part I. Unified Scalable Interpretation
Chapter 3. *ActiVis*: Visual Exploration of Industry-Scale Deep Learning Models M. Kahng, P. Andrews, A. Kalro, and D. H. Chau. *IEEE Transactions on Visualization and Computer Graphics, 24(1) (VAST 2017)*
Part II. Data-driven Model Auditing
Chapter 4. *MLCube*: Interactive Model Comparison with Data Cube Analysis M. Kahng, D. Fang, and D. H. Chau. *Workshop on Human-in-the-Loop Data Analytics (HILDA@SIGMOD 2016)*
Chapter 5. *FairVis*: Discovering Intersectional Bias in Machine Learning Á. Cabrera, W. Epperson, F. Hohman, M. Kahng, J. Morgenstern, and D. H. Chau. *IEEE Conference on Visual Analytics Science and Technology (VAST 2019)*
Part III. Learning Complex Models by Experimentation
Chapter 6. *GAN Lab*: Learning Deep Generative Models by Interactive Experimentation M. Kahng, N. Thorat, D. H. Chau, F. Viégas, and M. Wattenberg. *IEEE Transactions on Visualization and Computer Graphics, 25(1) (VAST 2018)*
Chapter 7. *ETable*: Interactive Browsing and Querying of Relational Databases M. Kahng, S. Navathe, J. Stasko, and D. H. Chau. *Proceedings of the VLDB Endowment (VLDB 2016)*

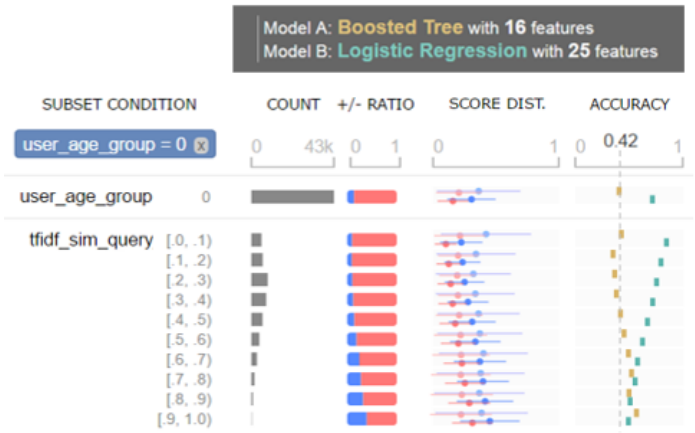


Figure 1.2: *MLCube* enables users to compare multiple models by slicing and dicing data instances based on their features.

1.2.2 Part II. Data-driven Model Auditing

While tools like *ActiVis* promote people's understanding of a model by visualizing how it responds to data instances and subsets, the interpretation of a model itself is only one part of many different tasks in applied machine learning. Building machine learning models involves several analytics stages (e.g., feature extraction, model selection) and often requires analysis of how input datasets affect results over a long machine learning pipeline. *How can we assist researchers and practitioners who work on various stages of a machine learning workflow, to identify potentially problematic data groups, so that they can discover insights and potentially fix the problems?*

***MLCube*: Interactive Model Comparison with Data Cube Analysis** (Chapter 4). One of the practical challenges in applied machine learning is to compare and select the best models among many candidates. We observed that users are often interested in how certain groups of data instances respond to different models. It motivated us to develop *MLCube* [88], a visual exploration tool for analyzing and comparing machine learning results by slicing and dicing them based on data features or attributes (Figure 1.2). Users can visually explore aggregate statistics and evaluation metrics over a large number of

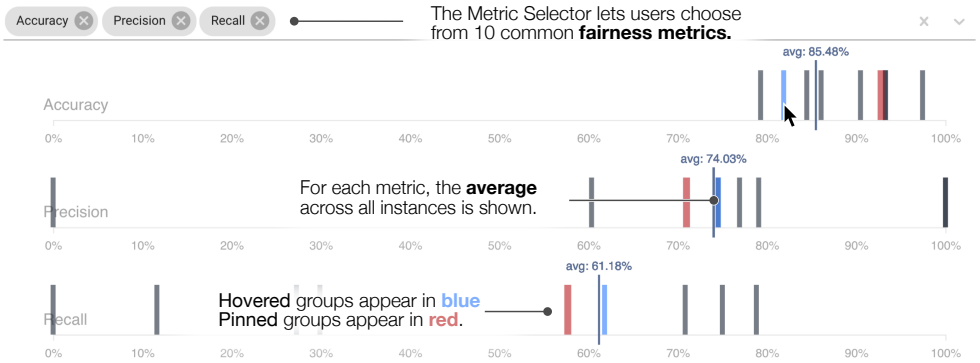


Figure 1.3: *FairVis* helps users discover bias in machine learning models by visualizing how different subgroups compare to one another according to various performance metrics.

subsets and interactively drill down into models using data cube operations. This can help users identify problematic data groups, spot interesting patterns, and inform their model selection process.

***FairVis*: Discovering Intersectional Bias in Machine Learning** (Chapter 5). Another important practical challenge recently identified by researchers is how to ensure machine learning models produce fair results for various population. Researchers have discovered that a machine learning model may produce much worse accuracy for certain groups of people (e.g., people of color) than for the overall population. This can result in many problems, as AI-powered systems continue to make important decisions across social domains. Thus, it is important to help data scientists better audit the fairness of their machine learning models. We developed *FairVis* [30], a novel visual analytics tool for users to discover intersectional bias in models. *FairVis*'s coordinated views allow users to explore a high-level overview of subgroup performances and subsequently drill down into detailed investigation of specific subgroups (Figure 1.3).

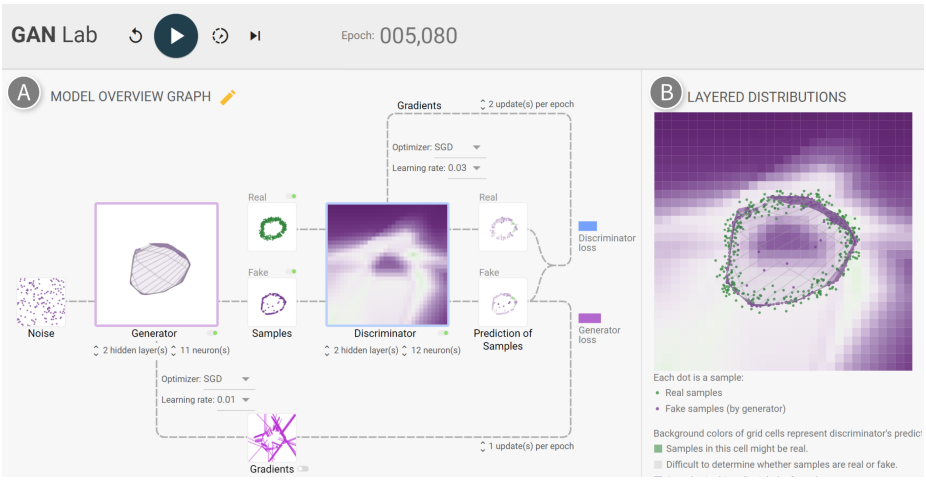


Figure 1.4: With *GAN Lab*, users can interactively create generative models and visually inspect how they generate data distributions.

1.2.3 Part III. Learning Complex Models by Experimentation

Recent success in deep learning has generated immense interest among practitioners and students, inspiring many to learn about this new technology. While visual and interactive approaches have been successfully developed to help these people learn AI and deep learning, most existing tools focus on simpler models. Modern deep learning models are becoming very complex and difficult for people to learn, introducing non-trivial challenges in designing visualization tools for them. *How can we broaden people's access to learning such complex models and making sense of complex structure of input datasets?*

***GAN Lab*: Learning Deep Generative Models by Interactive Experimentation** (Chapter 6). In collaboration with Google Brain's People+AI Research (PAIR) group, we built *GAN Lab* [90], a visual, interactive tool for learning Generative Adversarial Networks (GANs), one of the most popular, but hard-to-understand deep learning models. *GAN Lab* enables users to *interactively create GANs and experiment with different combinations of hyperparameters*, to learn how the models learn to generate data distributions (Figure 1.4). *GAN Lab* overcomes a major practical challenge in training deep learning models by using *TensorFlow.js*, an in-browser deep learning library, for users to use *GAN Lab* only with

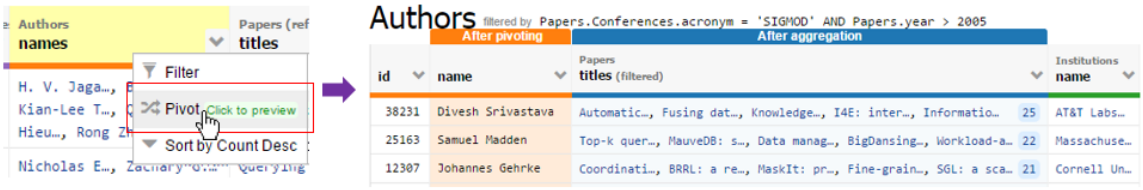


Figure 1.5: With *ETable*, users can easily browse multi-faceted data and interactively specify complex queries in relational databases.

their browsers without specialized backend. This implementation approach *significantly broadens people's education access* to interactive tools for modern deep learning technologies.

***ETable*: Interactive Browsing and Querying of Relational Databases** (Chapter 7). One of the important first steps in building machine learning models is making sense of input datasets. To help data analysts rapidly develop deep understanding of unfamiliar datasets stored in relational databases, the most popular type of databases, we developed *ETable* [89] for interactively browsing and navigating the complex relational structures of databases. *ETable*'s novel way of interactively constructing an extended table, based on *denormalization*, enables users to easily explore and query relational database tables (Figure 1.5). Our user studies indicate that users can construct complex queries much faster with *ETable* than a commercial, widely-used database administration tool.

1.3 Thesis Statement

Interactive visualization tools designed to provide a high-level overview of machine learning models and methods for drilling down into detailed explorations of how input datasets affect the models' results can help users (1) interpret large-scale models through data instances and subsets, (2) discover insights for selecting better-performing and unbiased models, and (3) learn the inner-workings of complex models by actively experimenting with them.

1.4 Research Contributions

My thesis makes research contributions through multiple major fronts.

- **New design principles**: My dissertation contributes novel design principles for developing interactive visualization tools for complex machine learning models. To help users make sense of the overall structure of models and perform detailed analysis, we designed our tools that provide users with *both a* ***high-level visual overview*** *of the models and* ***interactive methods to drill down into details*** *of the models or datasets*. In *ActiVis*, users can start their exploration with a graph-structured overview, and then dive into details of neurons' activation (Chapter 3). Also, *GAN Lab*'s coordinated views help users perform experimentations while visualizing a model's architecture (Chapter 6).

- **Novel data exploration models**: We contribute new ways to analyze how datasets affect machine learning results. The *MLCube* framework enables users to flexibly specify *data subset* by considering every part of a machine learning pipeline (Chapter 4). Through the powerful data cube framework, users can also interactively drill down into specific subsets, to perform more in-depth exploration. The *ActiVis* system further ***unifies the subset-level analysis with the instance-level analysis***, which allows to ***scale*** to large-scale datasets (Chapter 3). *ETable* also contributes new models for exploring data (Chapter 7).

- **New scalable, deployed systems**: We present new scalable systems for interpreting large-scale machine learning systems. *ActiVis*'s multiple scalable techniques enabled it to ***scale to industry-scale datasets and models*** and ***deploy*** to Facebook's internal machine learning platform (Chapter 3). *MLCube*'s scalable system design also led to a deployment by Facebook and influenced Google's open-source library (Chapter 4).

- **New broadly accessible approaches**: Our browser-based visualization tools signif-

icantly broaden public's access to modern AI technologies. *GAN Lab* overcomes a major practical challenge in deploying interactive tools for deep learning, by enabling users to *learn about models by playfully training and* ***experimenting*** *with them on web browser* (Chapter 6). *FairVis* also allows users to audit fairness of machine learning models on their browser (Chapter 5). Both tools have been open-sourced.

1.5 Impact

My research has made significant impact to society and industry.

- Our ***ActiVis*** system (Chapter 3) for interpreting deep learning models has been deployed on *FBLearner*, Facebook's internal machine learning platform that is used by more than 25% of Facebook's engineering team [55]. The paper for *ActiVis* [86] has been selected as one of the top four papers among 99 papers presented at the *IEEE VIS 2017* conference, to be invited to present at the *ACM SIGGRAPH 2018* conference. *ActiVis* is also patent-pending.

- Our ***MLCube*** framework (Chapter 4) for analyzing models by data slices has also been deployed on Facebook's internal machine learning platform. *MLCube*'s core idea has influenced the development of a Google's open-source system integrated into *TensorFlow*, the most popular deep learning library.

- Our ***GAN Lab*** tool (Chapter 6) for learning Generative Adversarial Networks has been open-sourced in collaboration with Google Brain's People+AI Research (PAIR) group, and recognized by many researchers and practitioners. The news about our release has been spreaded by 800 individuals (i.e., "retweet" in Twitter)[1], and the system website has been used by 70,000 people from over 160 countries over the world within the first year after release.

[1] `https://twitter.com/minsukkahng/status/1037016214575505409`

- My research has been recognized by both a **Google PhD Fellowship** and an **NSF Graduate Research Fellowship**. The former is one of the most prestigious industry-supported fellowship in computer science, and the latter is also a prestigious fellowship that supports outstanding graduate students, awarded by National Science Foundation.

CHAPTER 2

RELATED WORK

This chapter reviews related work. I first review work on using interactive visualization for interpreting machine learning results. Then I describe related work on visualization for the understanding of deep learning models. Lastly, I review interactive interpretation approaches that consider machine learning workflows.

2.1 Machine Learning Interpretation through Visualization

Machine learning interpretation. As the complexity of machine learning algorithms increases, many researchers have recognized the importance of model interpretability or explainability [149, 115, 52, 76]. While overall model accuracy has been primarily used to select models, machine learning engineers or researchers often want to understand why and when a model would perform better than others. This is important because without the understanding of models, they cannot trust the model and would not know how to further improve it.

Interactive visualization of results for a single instance. To help users better interpret machine learning models and their results, many interactive tools have been developed [123, 103, 113, 62, 170, 173, 43, 76]. In designing tools for explaining how machine learning models work, revealing relationships between data and models is one of the the most important design goals [142, 141]. A widely-used approach is helping users track how a model respond to an individual example (i.e., training or test instance), which

we call *instance-level analysis*. Kulesza et al. [102] presented an interactive system that explains how models made predictions for each instance. Amershi et al. [9] developed ModelTracker, a visualization tool that shows the distribution of instance scores for binary classification tasks and allows users to examine each instance individually. Squares [148] is an extension of ModelTracker for multi-classification tasks. Facets [181] is another interactive tool that visualizes instances, enabling users to browse a large number of instances by organizing them with their features or attributes.

Scaling to large datasets. While the instance-level exploration is helpful for tracking how models respond to individual examples, it does not easily scale to large datasets used in practice because a small number of sample examples has to be selected. To tackle this challenge, feature- or subset-level analysis has been used to explain the relationships between data and models, as machine learning features make it possible for instances to be grouped and sliced in multiple ways. Researchers have utilized *features* to visually describe how the models captured the structure of datasets [102, 100, 98, 27]. Kulesza et al. [102] used the importance weight of each feature in the Naive Bayes algorithm, and Krause et al. [100] used *partial dependence* to show the relationships between features and results.

Subset-level analysis. To enable users to analyze results not only by predefined features, researchers have developed tools that enable users to specify instance subsets. Specifying groups can often be a first step for analyzing machine learning results [101], as it provides users with an effective way for analyzing complex multidimensional data. In particular, people in the medical domain often perform similar processes, called *cohort construction*, and Krause et al. [101] developed an interactive tool that helps this process. McMahan et al. [126] presented a tool that allows users to visually compare the performance differences between models by subsets. Researchers have also used automated methods to find such subsets [97, 46, 129, 107]. Krause et al. [97] developed methods that find a set of binary features that can change prediction results if removed. Chung et al. [46] developed efficient methods to search for slices, combined with statistical tests to ensure

that the sizes of the slices are large enough. Our *ActiVis* (in Chapter 3) and *MLCube* (in Chapter 4) systems have been built on this line of research on subset-level analysis. *ActiVis* combines the subset-level analysis with the instance-level analysis and *MLCube* enables users to flexibly specify subsets and interactively explore a large number of subsets.

2.2 Visualization of Deep Learning Models

Visualization for conceptual understanding of deep learning. To help people, including those without computer science background, learn about deep learning, researchers and practitioners have written articles with visualizations and developed interactive tools accessible on the Web. One well-known example is Olah's series of essays,[1] explaining mathematical concepts behind deep learning using visualizations (e.g., on how neural networks transform and manipulate manifolds [137]). Another popular example is Karpathy's collection of web-based demos.[2] His demo for a convolutional neural network model [93] dynamically visualizes intermediate results, such as neuron activation. Olah's articles and Karpathy's demos have inspired many researchers to develop interactive visualizations for people to easily understand deep learning techniques [162, 72]. One example is *Deep Visualization Toolbox* [189] which visualizes activation information of convolutional neural networks for images instantly provided from webcam. Users can interactively see how results change depending on the input. Another notable example is *TensorFlow Playground* [162], an interactive visualization tool for non-experts to train simple neural network models and visualize internal components, such as neurons and weights. A new online interactive journal, *Distill*, has also been created, dedicated to interactive explanation of machine learning [138] which has featured many articles with interactive visualization [178, 64, 31, 139]. However, most existing visualizations for the conceptual understanding of deep learning

[1] Colah's blog, `http://colah.github.io`

[2] ConvNetJS Browser Demos, `https://cs.stanford.edu/people/karpathy/convnetjs/`

models focus on simpler models, and we present *GAN Lab* (in Chapter 6) designed for more complex models.

Visual analytics for deep learning models. With the growth of deep learning models, many visual analytics tools for deep learning have been developed, as we surveyed in [76]. One of the most widely-used tools, *TensorFlow Graph Visualizer* [183], visualizes model structures, to help researchers and engineers build mental models about them. Another popular way to interpreting deep learning models is visualizing high-dimensional representations of models, often called *embedding*, by projecting them into two dimensions [163, 147, 143] based on projecting algorithms like *t-Distributed Stochastic Neighbor Embedding (t-SNE)* [124]. *Embedding Projector* [163] is a popular tool that supports the visualization of embedding. Many other tools focus to visually summarize model results for interpreting how specific models respond to their datasets. For example, CNNVis [118] is designed for inspecting results from *convolutional neural networks (CNNs)*. This work models neurons as a directed graph and utilized several techniques, such as hierarchical clustering for grouping neurons and *bi-directional edge bundling* for summarizing edges among neurons. Several tools have been proposed for *recurrent neural networks (RNNs)* based models which are widely used for text data, which include LSTMVis [168], RNNVis [128], and Seq2Seq-Vis [167]. Tools for unsupervised generative models also exist. For instance, DGMTracker [117] allows experts to diagnose and monitor the training process of generative models through visualization of time-series data on data-flow graphs, and GANViz [177] helps experts evaluate and interpret trained results through multiple views, including one showing the distributions of real and fake image samples, for a selected epoch.

2.3 Interactive Analysis in Machine Learning Workflow

Importance of considering workflow. Machine learning systems used in practice often involve many different analytic stages (e.g., data preprocessing, feature selection, and model debugging) [157, 144]. Therefore, in developing interactive tools for supporting machine learning practitioners, it is important to design tools that consider their machine learning workflow. Patel et al. [141] presented a development environment for developers to implement classification models and argued that interactive tools that support the entire machine learning process can accelerate the understanding of models. Sculley et al. [157] also argued that the bottleneck of practical machine learning systems is often caused by the lack of data dependency over the machine learning pipelines. The database community acknowledges the importance of managing data flow. With this expertise, many researchers have studied on helping machine learning engineers perform *feature engineering* [10, 192], and researchers have also studied on model selection or interpretation [104, 38]. Chen et al. [38] developed a *prediction cube* framework to examine the effect of features using data cube analysis. Our *MLCube* framework (in Chapter 4) advances prior work by allowing users to specify subsets over any intermediate data produced throughout the machine learning pipeline from input datasets to output metrics.

Analysis for identifying problems and improving models. Building machine learning models is an iterative process [142, 187, 54, 8]. Engineers often keep trying to refine, improve, or debug their existing models to make it perform better. However, model debugging is a big challenge in machine learning, especially for deep learning models. While machine learning researchers focus on refining or modifying hyperparameters of models, one of the practically effective approaches is identifying problems in datasets [144], and many visual or interactive analytics systems have been proposed to address this problem. This includes identifying misclassified instances [129], errors in labels [39], and fairness-related issues (e.g. *FairVis* in Chapter 5). Interactive tools that support the identification of

problems can also lead to improving performance [180, 167, 129, 136, 185, 179]. For example, the *What-If* tool [180], which enables users to generate and test hypothesis, can be used to look for counterfactual explanations [106] for specific instances and also to modify a classifier's threshold to change which fairness principles are being satisfied.

PART I

UNIFIED SCALABLE INTERPRETATION

Overview

Deep learning models often work with very large datasets, introducing many non-trivial design challenges in developing interactive visualization tools for interpreting such models. While a common approach to interpreting machine learning models, called instance-level analysis, helps users explore a model's response for an individual instance, it does not easily scale to large datasets. The first part of my dissertation describes my new approach to interpreting deep learning models that use large data by unifying the instance-level and subset-level analyses. In particular, this part describes the following work:

- *ActiVis* (Chapter 3) designed for interpreting industry-scale deep neural network models and deployed by Facebook.

CHAPTER 3

ACTIVIS: VISUAL EXPLORATION OF INDUSTRY-SCALE DEEP LEARNING MODELS

Despite the recent interest in developing visual tools to help users interpret deep learning models, the complexity and wide variety of models deployed in industry, and the large-scale datasets that they used, pose unique design challenges that are inadequately addressed by existing work. This chapter describes *ActiVis*, a visual analytics system for interpreting large-scale deep learning models and results, developed through participatory design sessions with over 15 researchers and engineers at Facebook. By tightly integrating multiple coordinated views, such as a *computation graph* overview of the model architecture, and a *neuron activation* view for pattern discovery and comparison, users can explore complex deep neural network models at both the instance- and subset-level. *ActiVis* has been deployed on Facebook's machine learning platform. We present case studies with Facebook researchers and engineers, and usage scenarios of how *ActiVis* may work with different models.

3.1 Introduction

Deep learning has led to major breakthroughs in various domains, such as computer vision, natural language processing, and healthcare. Many technology companies, like Facebook,

This chapter is adapted from work appeared at IEEE VAST 2017 [86].

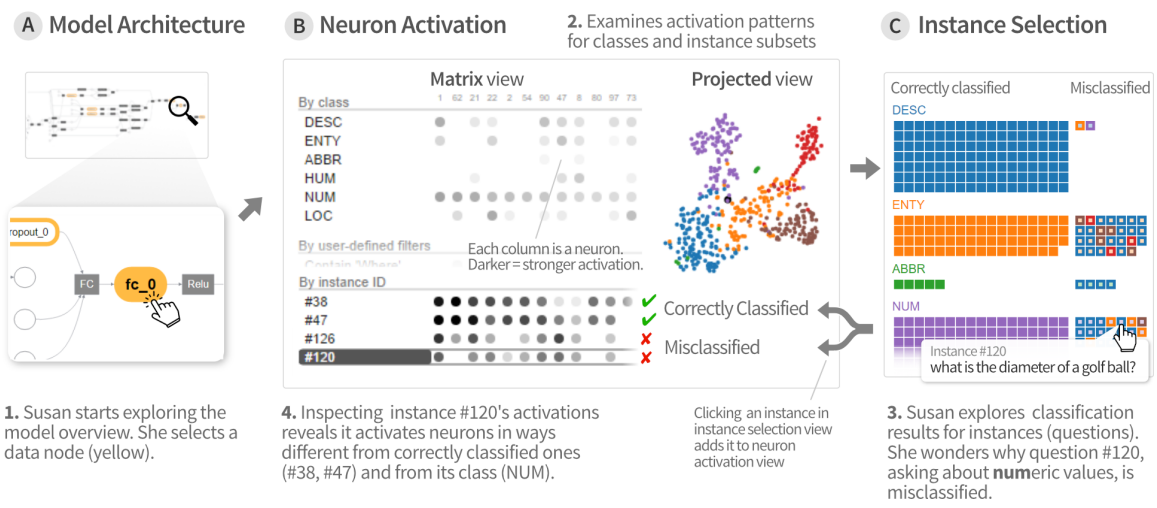


Figure 3.1: *ActiVis* integrates several coordinated views to support exploration of complex deep neural network models, at both instance- and subset-level. **1.** Our user Susan starts exploring the model architecture, through its *computation graph* overview (at A). Selecting a *data node* (in yellow) displays its *neuron activations* (at B). **2.** The *neuron activation matrix view* shows the activations for instances and instance subsets; the *projected view* displays the 2-D projection of instance activations. **3.** From the *instance selection* panel (at C), she explores individual instances and their classification results. **4.** Adding instances to the matrix view enables comparison of activation patterns across instances, subsets, and classes, revealing causes for misclassification.

have been increasingly adopting deep learning models for their products [3, 2, 47]. While powerful deep neural network models have significantly improved prediction accuracy, understanding these models remains a challenge. Deep learning models are more difficult to interpret than most existing machine learning models, because they capture nonlinear hidden structures of data using a huge number of parameters. Therefore, in practice, people often use them as "black boxes", which could be detrimental because when the models do not perform satisfactorily, users would not understand the causes or know how to fix them [102, 149].

Despite the recent increasing interest in developing visual tools to help users interpret deep learning models [118, 189, 45, 163], the complexity and wide variety of models deployed in industry, and the large-scale datasets that they use, pose unique challenges that

are inadequately addressed by existing work. For example, deep learning tasks in industry often involve different types of data, including text and numerical data; however most existing visualization research targets image datasets [189]. Furthermore, in designing interpretation tools for real-world use and deployment at technology companies, it is a high priority that the tools be flexible and generalizable to the wide variety of models and datasets that the companies use for their many products and services. These observations motivate us to design and develop a visualization tool for interpreting industry-scale deep neural network models, one that can work with a wide range of models, and can be readily deployed on Facebook's machine learning platform.

Through participatory design with researchers, data scientists, and engineers at Facebook, we have identified common analysis strategies that they use to interpret machine learning models. Specifically, we learned that both **instance-** and **subset-based** exploration approaches are common and effective. Instance-based exploration (e.g., how individual instances contribute to a model's accuracy) have demonstrated success in a number of machine learning tasks [102, 9, 141]. As individual instances are familiar to users, exploring by instances accelerates model understanding. Another effective strategy is to leverage input features or instance subsets specified by users [100, 102]. Slicing results by features helps reveal relationships between data attributes and machine learning algorithms' outputs [126, 88, 141]. Subset-based exploration is especially beneficial when dealing with huge datasets in industry, which may consist of millions or billions of data points. Interpreting model results at a higher, more abstract level helps drive down computation time, and help user develop general sense about the models.

Our tool, called ***ActiVis***, aims to support both interpretation strategies for visualization and comparison of multiple instances and subsets. *ActiVis* is an interactive visualization system for deep neural network models that (1) unifies instance- and subset-level inspections, (2) tightly integrates overview of complex models and localized inspection, and (3) scales to a variety of industry-scale datasets and models. *ActiVis* visualizes how

neurons are activated by user-specified instances or instance subsets, to help users understand how a model derives its predictions. Users can freely define subsets with raw data attributes, transformed features, and output results, enabling model inspection from multiple angles. While many existing deep learning visualization tools support instance-based exploration [72, 189, 45, 163], *ActiVis* is the first tool that simultaneously supports instance- and subset-based exploration of the deep neural network models. In addition, to help users get a high-level overview of the model, *ActiVis* provides a graph-based representation of the model architecture, from which the user can drill down to perform localized inspection of activations at each model layer (node).

Illustrative scenario. To illustrate how *ActiVis* works in practice, consider our user Susan who is training a word-level *convolutional neural network* (CNN) model [95] to classify question sentences into one of six categories (e.g., whether a question asks about *numeric* values, as in *"what is the diameter of a golf ball?"*). Her dataset is part of the TREC question answering data collections[1] [112].

Susan is new to using this CNN model, so she decides to start by using its default training parameters. After training completes, she launches *ActiVis*, which runs in a web browser. *ActiVis* provides an overview of the model by displaying its architecture as a computation graph (Figure 3.1A, top), summarizing the model structure. By exploring the graph, Susan learns about the kind of operations (e.g., convolution) that are performed, and how they are combined in the model.

Based on her experience working with other deep learning models, she knows that a model's performance is strongly correlated with its last hidden layer, thus it would be informative to analyze that layer. In *ActiVis*, a layer is represented as a rounded rectangular *node* (highlighted in yellow, in Figure 3.1A, bottom).

Susan clicks the node for the last hidden layer, and *ActiVis* displays the layer's *neuron activation* in a panel (Figure 3.1B): the *neuron activation matrix view* on the left shows

[1] `http://cogcomp.cs.illinois.edu/Data/QA/QC/`

how neurons (shown as columns) respond to instances from different classes (rows); and the *projected view* on the right shows the 2-D projection of instance activations.

In the *matrix view*, stronger neuron activations are shown in darker gray. Susan sees that the activation patterns for the six classes (rows) are quite visually distinctive, which may indicate satisfactory classification. However, in the *projected view*, instances from different classes are not clearly separated, which suggests some degree of misclassification.

To examine the misclassified instances and to investigate why they are mislabeled, Susan brings up the *instance selection panel* (Figure 3.1C). The classification results for the **NUM**ber class alarm Susan, as many instances in that class are misclassified (shown in right column). She examines their associated question text by mouse-overing them, which shows the text in popup tooltips. She wants to compare the activation patterns of the correctly classified instances with those of the misclassified. So she adds two correct instances (#38, #47) and two misclassified instances (#120, #126) to the *neuron activation matrix view* — indeed, their activation patterns are very different (Figure 3.1.4).

Taking a closer look at the *instance selection panel*, Susan sees that many instances have blue borders, meaning they are misclassified as **DESC**ription. Inspecting the instances' text reveals that they often begin with *"What is"*, which is typical for questions asking for descriptions, though they are also common for other question types, as in *"What is the diameter of a golf ball?"* which is a numeric question (Figure 3.1.3).

To understand the extent to which instances starting with *"What is"* are generally misclassified by the model, Susan creates an *instance subset* for them, and *ActiVis* adds this subset as a new row in the *neuron activation matrix view*. Susan cannot discern any visual patterns from the subset's seemingly scattered, random neuron activations, suggesting that the model may not yet have learned effective ways to distinguish between the different intents of *"What is"* questions. Based on this finding, she proceeds to train more models with different parameters (e.g., consider longer n-grams) to better classify these questions.

ActiVis integrates multiple coordinated views to enable Susan to work with complex

models, and to flexibly explore them at instance- and subset-level, helping her discover and narrow in to specific issues.

Deployment. *ActiVis* has been deployed on the machine learning platform at Facebook. A developer can visualize a deep learning model using *ActiVis* by adding only a few lines of code, which instructs the model's training process to generate data needed for *ActiVis*. *ActiVis* users at Facebook (e.g., data scientists) can then train models and use *ActiVis* via *FBLearner Flow* [55, 12], Facebook's internal machine learning web interface, without writing any additional code.

ActiVis's main contributions include:

- A novel visual representation that unifies instance- and subset-level inspections of neuron activations, which facilitates comparison of activation patterns for multiple instances and instance subsets. Users can flexibly specify subsets using input features, labels, or any intermediate outcomes in a machine learning pipeline (Section 3.3.2).
- An interface that tightly integrates an overview of graph-structured complex models and local inspection of neuron activations, allowing users to explore the model at different levels of abstraction (Section 3.3.3).
- A deployed system scaling to large datasets and models (Section 3.3.4).
- Case studies with Facebook engineers and data scientists that highlight how *ActiVis* helps them with their work, and usage scenarios that describe how *ActiVis* may work with different models (Section 3.5).

3.2 Analytics Needs for Industry-Scale Problems

The *ActiVis* project started in April 2016. Since its inception, we have conducted participatory design sessions with over 15 Facebook engineers, researchers, and data scientists

across multiple teams to learn about their visual analytics needs. Together, we collaboratively design and develop *ActiVis* and iteratively improve it.

In Section 3.2.1, we describe the workflow of how machine learning models are typically trained and used at Facebook, and how results are interpreted. This discussion provides the background information and context for which visualization tools may help improve deep learning model interpretation.

In Section 3.2.2, we summarize our main findings from our participatory design sessions to highlight six key design challenges that stem from Facebook's needs to work with large-scale datasets, complex deep learning model architectures, and diverse analytics needs. These challenges have been inadequately addressed by current deep learning visualization tools, and they motivate and shape our design goals for *ActiVis*, which we will describe in Section 3.3.1.

3.2.1 Background: Machine Learning Practice at Facebook

Facebook uses machine learning for some of their products. Researchers, engineers, and data scientists from different teams at Facebook perform a wide range of machine learning tasks.

We first describe how Facebook's machine learning platform helps users train models and interpret their results. Then, we present findings from our discussion with machine learning users and their common analytics patterns in interpreting machine learning models. These findings guide our discovery of design challenges that *ActiVis* aims to address.

FBLearner Flow: Facebook's Machine Learning Platform

To help engineers, including non-experts of machine learning, to more easily reuse algorithms in different products and manage experiments with ease, Facebook built a unified machine learning platform called *FBLearner Flow* [55, 12]. It supports many machine learning workflows. Users can easily train models and see their results using the FBLearner

Flow interface without writing any code. For example, users can train a model by picking a relevant workflow from a collection of existing workflows and specifying several input parameters for the selected workflow (e.g., location of training dataset, learning parameters). The FBLearner Flow interface is particularly helpful for users who want to use existing machine learning models for their datasets without knowing their internal details.

Once the training process is done, the interface provides high-level information to aid result analysis (e.g., precision, accuracy). To help users interpret the results from additional multiple aspects, several other statistics are available in the interface (e.g., partial dependence plots). Users can inspect models' internal details via interactive visualization (e.g., for decision trees) [12]. As deep neural network models gain popularity, developing visualization for their interpretation is a natural step for FBLearner Flow.

Analytics Patterns for Interpretation

To better understand how machine learning users at Facebook interpret model results, and how we may design *ActiVis* to better support their analysis, we conducted participatory design sessions with over 15 engineers and data scientists who regularly work with machine learning and deep neural network models. At the high level, we learned that instance- and subset-based strategies are both common and effective, echoing findings from existing research.

Instance-based analysis. One natural way for users at Facebook to understand complex models is by tracking how an individual example (i.e., training or test instance) behaves inside the models; users often have their own collection of example instances, for which they know their characteristics and ground truth labels. Instance-level exploration is especially useful when an instance is easy to interpret. For example, an instance consisting of text only is much easier to understand than an instance consisting of thousands of numerical features extracted from an end user's data.

Subset-based analysis. Instance-based analysis, however, is insufficient for all cases.

Inspecting instances individually can be tedious, and sometimes hinder insight discovery, such as when instances are associated with many hard-to-interpret numerical features. We learned that some Facebook researchers find subset-based analysis to be more helpful for their work. For example, suppose an instance represents an article that consists of many numerical features extracted from its attributes (e.g., length, popularity). Some users would like to understand how the models behave at higher-level categorization (e.g., by topic, publication date). In addition, some users have curated instance subsets. Understanding model behavior through such familiar subsets promotes their understanding.

3.2.2 Design Challenges

Besides reaffirming the importance of two analysis strategies discussed above, and the need to support them simultaneously in *ActiVis*, we have identified additional design challenges through the participatory design sessions. We summarize them into six key design challenges. Thus far, they have not been adequately addressed by existing deep learning visualization tools. And they shape the main design goals of *ActiVis*, which we will describe in Section 3.3.1.

We have labeled the six challenges C1 – C6 and have grouped them into three categories with the labels *data*, *model*, and *analytics*, which indicate the causes for which the challenges arise.

C1. **Diverse input sources and formats** [DATA]

While deep learning has become popular because of its superior performance for image data, it has also been applied to many different data formats, including text and numerical features [95, 85, 3, 47]. Furthermore, a single model may jointly use multiple types of data at a time. For example, to classify a Facebook post, a model may jointly leverage its textual content, attached photos, and user information, each of which may be associated with many data attributes [3]. Working with such variety of data sources and formats opens up many opportunities for model interpretation; for

example, we may be able to more easily categorize instances using their associated numerical features that can be more readily understood, instead of going the harder route of using image-based features.

C2. **High data volume** [DATA]

Facebook, like many other companies, has a large amount of data. The size of training data often reaches billions of rows and thousands of features. This sheer size of data render many existing visualization tools unusable as they are often designed to visualize the whole dataset.

C3. **Complex model architecture** [MODEL]

Many existing visualization tools for deep learning models often assume simple linear architectures where data linearly flow from the input layer to the output layer (e.g., a series of convolution and max-pooling layer in AlexNet) [189, 118, 45]. However, most practical model architectures deployed in industry are very complex [47]; they are often deep and wide, consisting of many layers, neurons, and operations.

C4. **A great variety of models** [MODEL]

Researchers and engineers at Facebook develop and evaluate models for products every day. It is important for visualization tools to be generalizable so they can work with many different kinds of models. A visualization system would likely be impractical to use or to deploy if a small change to a model requires significant changes made to existing code or special case handling.

C5. **Diverse subset definitions** [ANALYTICS]

When performing subset-based analysis, users may want to define subsets in many different ways. Since there are a large number of input formats and input features, there are numerous ways to specify subsets. Instead of providing a fixed set of ways to define subsets, it is desirable to make this process flexible so that users can flexibly define subsets that are relevant to their tasks and goals.

C6. **Simultaneous need for performing instance- and subset-level analysis** [ANALYTICS] Instance- and subset-based are complementary analytics strategies, and it is important to support both at the same time. Instance-based analysis helps users track how an individual instance behaves in the models, but it is tedious to inspect many instances one by one. By specifying subsets and enabling their comparison with individual instances, users can learn how the models respond to many different slices of the data.

3.3 *ActiVis*: Visual Exploration of Neural Networks

Through the design challenges we identified (in Section 3.2.2) in our participatory design sessions with researchers, engineers, and data scientists at Facebook, we design and develop *ActiVis*, a novel interactive visual tool for exploring a wide range of industry-scale deep neural network models. In this section, we first present three main design goals distilled from our conversations with Facebook participants (Section 3.3.1). Then, for each design goal, we elaborate on how *ActiVis* achieves it through its system design and visual exploration features (Sects. 3.3.2-3.3.4). We label the three design goals G1 – G3.

3.3.1 Design Goals

G1. **Unifying instance- and subset-based analysis to facilitate comparison of multiple instance activations.** From our participatory design sessions, we learned that both instance- and subset-based analysis are useful and complementary. We aim to support subset-level exploration by enabling users to flexibly define instance subsets for different data types (C1, C5), e.g., a set of documents that contain a specific word. Subset-based analysis also allows users to explore datasets at higher-level abstraction, scaling to billion-scale data or larger (C2). Furthermore, we would like to unify instance- and subset-level inspections to facilitate comparison of multiple

instances and groups of instances in a single view (C6).

G2. **Tight integration of overview of model architecture and localized inspection of activations.** Industry-scale deep neural network models are often very complex, consisting of many operations (C3). Visualizing every detail and activation value for all intermediate layers can overwhelm users. Therefore, we aim to present the architecture of the models as a starting point of exploration, and let users switch to the detailed inspection of activations.

G3. **Scaling to industry-scale datasets and models through flexible system design.** For *ActiVis* to work with many different large-scale models and datasets used in practice, it is important for the system to be flexible and scalable. We aim to support as many different kinds of data types and classification models as what FBLearner currently does (e.g., image, text, numerical) (C1, C4). We would like to achieve this by developing a flexible, modularized system that allows developers to use *ActiVis* for their models with simple API functions, while addressing visual and computational scalability challenges through a multipronged approach (C2, C3).

3.3.2 Exploring Neuron Activations by Instance Subsets

Drawing inspiration from existing visualizations [189, 72, 118], *ActiVis* supports the visualization for individual instances. However, it is difficult for users to spot interesting patterns and insights if he can only visualize one instance at a time. For example, consider a hidden layer consisting of 100 neurons. The neuron activations for an instance is a 100-dimension vector consisting of 100 numerical values, where each element in the vector does not have any specific meaning. Instead, if multiple vectors of activation values are presented together, the user may more readily derive meaning by comparing them. For example, users may find that some dimensions may respond more strongly to certain instances, or some dimensions are negatively correlated with certain classes.

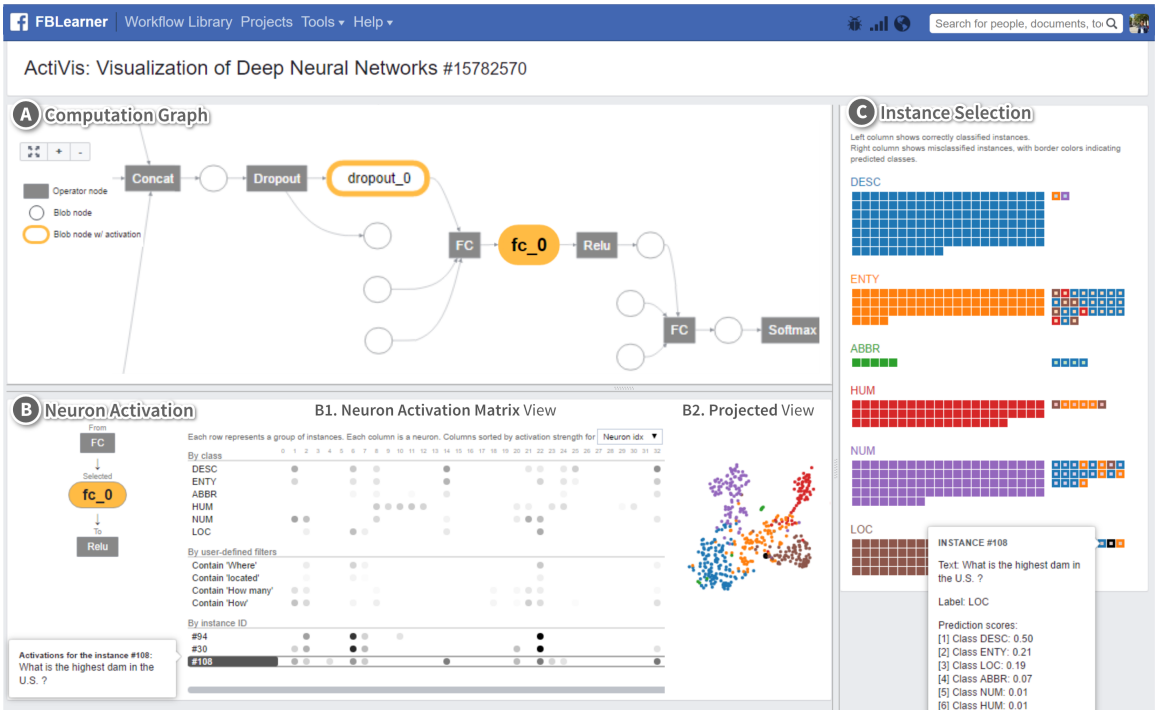


Figure 3.2: *ActiVis* integrates multiple coordinated views. **A.** The *computation graph* summarizes the model architecture. **B.** The *neuron activation* panel's *matrix view* displays activations for instances, subsets, and classes (at B1), and its *projected view* shows a 2-D t-SNE projection of the instance activations (at B2). **C.** The *instance selection* panel displays instances and their classification results; correctly classified instances shown on the left, misclassified on the right. Clicking an instance adds it to the neuron activation matrix view. The dataset used is from the public TREC question answering data collections [112]. The trained model is a word-level convolutional model based on [95].

A challenge in supporting the comparison of multiple instances stems from the sheer size of data instances; it is impossible to present activations for all instances. To tackle this challenge, we enable users to define *instance subsets*. Then we compute the average activations for instances within the subsets. The vector of average activations for a subset can then be placed next to the vectors of other instances or subsets for comparison.

The *neuron activation matrix*, shown at Figure 3.2B.1, illustrates this concept of comparing multiple instances and instance subsets, using the TREC question classification dataset[2] [112]. The dataset consists of 5,500 question sentences and each sentence is la-

[2] `http://cogcomp.cs.illinois.edu/Data/QA/QC/`

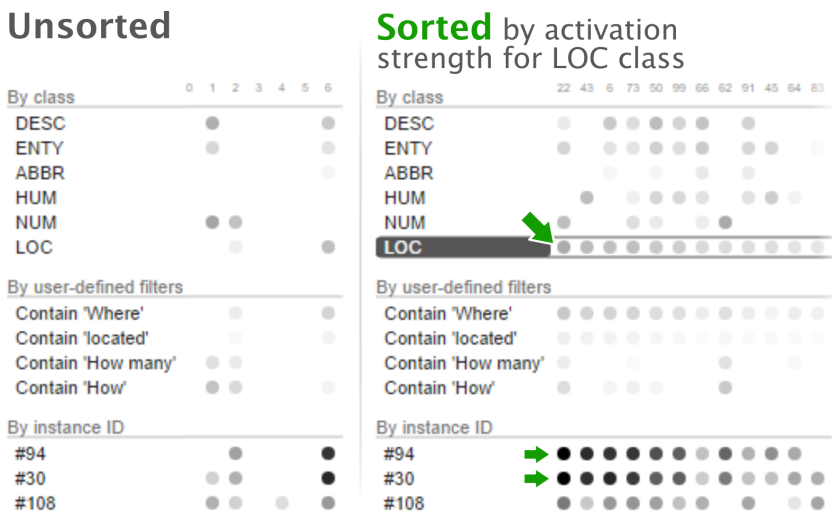


Figure 3.3: Sorting neurons (columns) by their average activation values for the *LOC* (location) class helps users more easily spot instances whose activation patterns are positively correlated with that of the class, e.g., instances #94 and #30 (see green arrows).

beled by one of six categories (e.g., is a question asking about *location*?). Figure 3.2B shows the activations for the last hidden layer of the word-level CNN model [95, 26]. Each row represents either an instance or a subset of instances. For example, the first row represents a subset of instances whose true class is 'DESC' (descriptions). Each column represents a neuron. Each cell (circle) is a neuron activation value for a subset. A darker circle indicates stronger activation. This matrix view exposes the hidden relationships between neurons and data. For instance, a user may find out a certain neuron is highly activated by instances whose true class is 'LOC'.

Flexible subset definition. In *ActiVis*, users can flexibly define instance subsets. A subset can be specified using multiple properties of the instances, in many different ways. Example properties include raw data attributes, labels, features, textual content, output scores, and predicted label. Our datasets consist of instances with many features and a combination of different types of data. Flexible subset definition enables users to analyze models from different angles. For example, for instances representing text documents, the user may create a subset for documents that contains a specific phrase. For instances

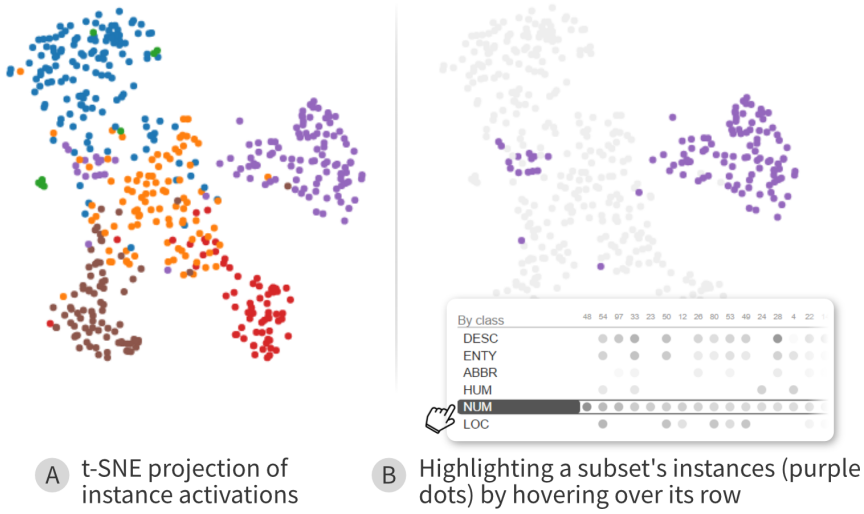


Figure 3.4: Hovering over an instance subset (e.g., for the NUMber class) highlights its instances (purple dots) in the t-SNE projected view.

containing numerical features, users can specify conditions, using operations similar to relational selections in databases (e.g., `age > 20`, `topic = 'sports'`). By default, a subset is created for each class (e.g., a subset for the 'DESC' class).

Sorting to reveal patterns. The difficulty in recognizing patterns increases with the number of neurons. *ActiVis* allows users to sort neurons (i.e., columns) by their activation values. For example, in Figure 3.3, the neurons are sorted based on the average activation values for the class 'LOC'. Sorting facilitates activation comparison and helps reveal patterns, such as spotting instances that are positively correlated with their true class in terms of the activation pattern (e.g., instances #94 and #30 correlate with the 'LOC' class in Figure 3.3).

2-D projection of activations. To help users visually examine instance subsets, *ActiVis* provides a 2-D *projected view* of instance activations. Projection of high-dimensional data into 2-D space has been considered an effective exploration approach [147, 163, 45, 43]. *ActiVis* performs *t-distributed stochastic neighbor embedding (t-SNE)* [124] of instance activations. Figure 3.2B.2 shows an example where each dot in the view represents an instance (colored by its true class), and instances with similar activation values are placed

closer together by t-SNE.

The projected view complements with the *neuron activation matrix* view (Figure 3.2B.1). Hovering over a subset's row in the matrix would highlight the subset's instances in the projected view (as shown in Figure 3.4), allowing the user to see how instances within the subsets are distributed. In the projected view, hovering over an instance would display its activations; clicking that instance will add it to the matrix view as a new row.

3.3.3 Interface: Tight Integration of Model, Instances, and Activation

The above visual representation of activations is the core of our visual analytics system. To help users interactively specify where to start their exploration of a large model, we designed and developed an integrated system interface. As depicted in Figure 3.2, the interface consists of multiple panels. We describe each of them below.

A: Overview of Model Architecture

Deep learning models often consist of many operations, which makes it difficult for users to fully understand their structure. We aim to provide an overview of the model architecture to users, so they can first make sense of the models, before moving on to parts of the models that they are interested in.

Deep neural network models are often represented as computation graphs (DAGs) (as in many deep learning frameworks like Caffe2[3], TensorFlow [2], and Theano[4]). The frameworks provide a set of operators (e.g., convolution, matrix multiplication, concatenation) to build machine learning programs, and model developers (who create new machine learning workflows for FBLearner Flow) write the programs using these building blocks. Presenting this graph to users would help them first understand the structure of the models and find

[3]Caffe2, `https://caffe2.ai/`
[4]Theano, `http://deeplearning.net/software/theano/`

interesting layers to explore the detailed activations.

There are several possible ways in visualizing computation graphs. One approach is to represent operators as nodes and variables as edges. This approach has gained popularity, thanks to its adoption by TensorFlow. Another way is to consider both an operator and a variable as a single node. Then the graph becomes a bipartite graph: the direct neighbors of an operator node are always variable nodes; the neighbors of a variable node are always operator nodes. Both approaches have their pros and cons. While the first approach can have a compact representation by reducing the number of nodes, the second one, a classical way to represent programs and diagrams, makes it easier to track data. For *ActiVis*, it would be better to make variable nodes easy to locate as we present activations for a selected variable. Therefore, we decided to represent the graph using the second approach.

The visualization of the computation graph is shown on the top panel (Figure 3.2A). The direction of data flow is from left (input) to right (output). Each node represents either an operator (dark rectangle) or tensor (circle). To explore this medium-sized graph (often $>$100 nodes), users can zoom and pan the graph using a mouse. When users hover over a node, its full name is shown, and when they click it, its corresponding activation is shown in the neuron activation panel.

B: Activation for Selected Node

When users select a node of interest from the computation graph, the corresponding neuron activation panel (Figure 3.2B) will be added to the bottom of the computation graph panel. The neuron activation panel has three subpanels: (0) the names of the selected node and its neighbors, (1) the neuron activation matrix view, and (2) the projected view. The left subpanel shows the name of the selected variable node and its neighbors. Users can hover over a node to highlight where it is located in the computation graph on the top. The neuron matrix view (Figure 3.2B.1) and projected view (Figure 3.2B.2) show instance activations for the selected node. Note that we described these views in Section 3.3.2.

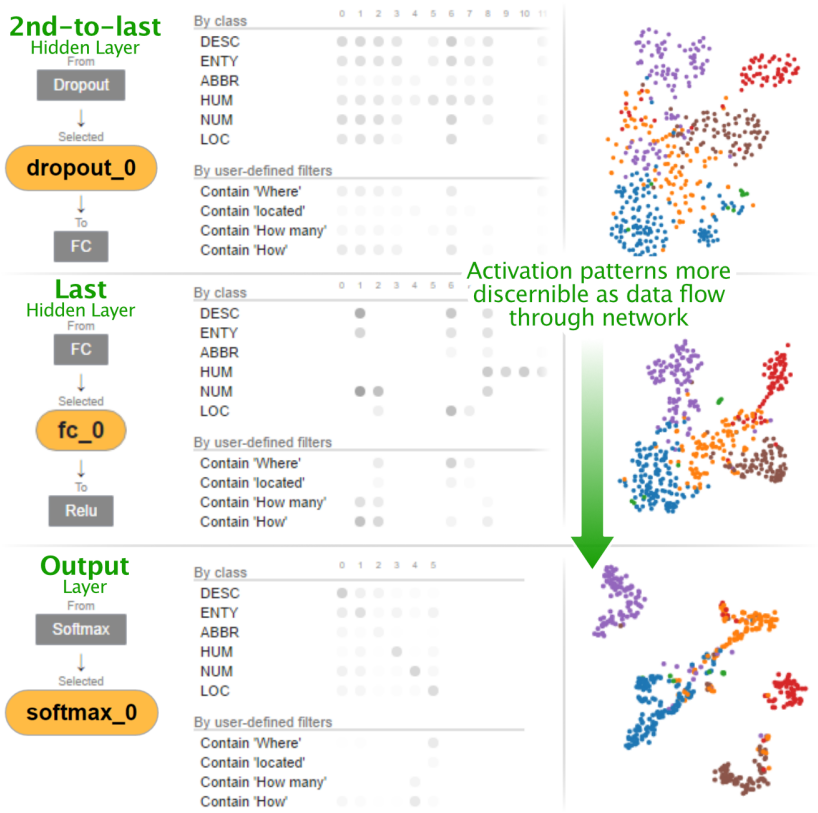


Figure 3.5: Users can simultaneously visualize and compare multiple layers' activations. Shown here, from top to bottom, are: the second-to-last hidden layer, the last hidden layer, and the output layer. Their projected views show that as instances flow through the network from input (top) to output (bottom), their activation patterns gradually become more discernible and clustered (in projected view).

Users can select multiple nodes and visually compare their activation patterns. Figure 3.5 illustrates that users can visually explore how models learned the hidden structure of data through multiple layers. The figure shows three layers, from top to bottom: the second-to-last hidden layer which concatenates multiple maxpool layers [95], the last hidden layer, and the output layer. As shown in the figure, the layer's projected views show that as data flow through the network, from input (top) to output (bottom), neuron activation patterns gradually become more discernible and clustered.

C: Instance Selection

The instance selection panel helps users get an overview of instances with their prediction results and determine which ones should be added to the neuron activation view for further exploration and comparison.

The panel is located at the right side on the interface. It visually summarizes prediction results. Each square represents an instance. Instances are vertically grouped based on their true label. Within a true label (row group), the left column shows correctly classified instances, sorted by their prediction scores in descending order (from top to bottom, and left to right within each row). The right column shows misclassified instances. An instance's fill color represents its true label, its border color the predicted label. When the user hovers over an instance, a tooltip will display basic information about the instance (e.g., textual content, prediction scores).

The panel also helps users determine which instances can be added to the activation view for further exploration. By hovering over one of the instance boxes, users can see the instance's activations. A new row is added to the activation view presenting the activation values for the selected instance. When users' mouse leaves the box, the added row disappears. To make a row persistent, users can simply click the box. In a similar fashion, users can add many rows by clicking the instance boxes. Then, they can compare activations for multiple instances and also compare those for instances with those for groups of instances.

3.3.4 Deploying *ActiVis*: Scaling to Industry-scale Datasets and Models

We have deployed *ActiVis* on Facebook's machine learning platform. Developers who want to use *ActiVis* for their model can easily do so by adding only a few lines of code, which instructs their models' training process to generate information needed for *ActiVis*'s visualization. Once model training has completed, the FBLearner Flow interface provides the user with a link to *ActiVis* to visualize and explore the model. The link opens in a new web

browser window.

ActiVis is designed to work with classification tasks that use deep neural network models. As complex models and large datasets are commonly used at Facebook, it is important that *ActiVis* be scalable and flexible, so that engineers can easily adopt *ActiVis* for their models. This section describes our approaches to building and deploying *ActiVis* on FBLearner, Facebook's machine learning platform.

Generalizing to Different Models and Data Types

One of our main goals is to support as many different kinds of data types and models as what FBLearner currently does (e.g., images, text, numerical). The key challenge is to enable existing deployed models to generate data needed for *ActiVis* with as little modification as possible. Without careful thinking, we would have to add a large amount of model-specific code, to enable *ActiVis* to work with different models. To tackle this challenge, we modularize the data generation process and define API functions for model developers so that they can simply call them in their code, to activate *ActiVis* for their models. In practice, for a developer to use *ActiVis* for a model, only three function calls are needed to be added (i.e., calling the *preprocess*, *process*, and *postprocess* methods). For example, developers can specify a list of variable nodes that users can explore, as an argument of the *preprocess* function (described in detail in Section 3.3.4). Furthermore, developers can leverage *user-defined functions* to specify how subsets are defined in *ActiVis*, a capability particularly helpful for the more abstract, unstructured data types, such as image and audio. For example, developers may leverage the output of an object recognition algorithm that detects objects (e.g., cats, dogs) to define image subsets (e.g., subset of images that contain dogs).

Scaling to Large Data and Models

ActiVis addresses visual and computational scalability challenges through multiple complementary approaches. Some of them were introduced in earlier sections (e.g., Section 3.3.2), such as *ActiVis*'s overarching subset-based analysis, and the simultaneous use of *neuron matrix* (for individual neuron inspection) and *projected view* (in case of many neurons). We elaborate on some of our other key ideas below.

Selective precomputation for variable nodes of interest. Industry-scale models often consist of a large number operations (i.e., variable nodes), up to hundreds. Although any variable node can be visualized in the activation visualization, if we compute activations for all of them, it will require significant computation time and space for storing the data. We learned from our discussion with experts and design sessions with potential users that it is typical for only a few variable nodes in a model to be of particular interest (e.g., last hidden layer in CNN). Therefore, instead of generating activations for all variable nodes, we let model developers specify their own default set of variable nodes. The model developers can simply specify them as an argument of the *preprocess* method. To explore variable nodes not included in the default set, a user can add them by specifying the variable nodes in the FBLearner Flow interface. Such nodes will then be available in the computation graph (highlighted in yellow).

User-guided sampling and visual instance selection. For billion-scale datasets, it is undesirable to display all data points in the instance selection panel. Furthermore, we learned from our design sessions that researchers and engineers are primarily interested in a small number of representative examples, such as "test cases" that they have curated (e.g., instances that should be labeled as Class 'LOC' by all well-performing models). To meet such needs, by default, we present a sample of instances in the interface (around 1,000), which meet the practical needs of most Facebook engineers. In addition, users may also guide the sampling to include arbitrary examples that they specify (e.g., their test cases).

Computing neuron activation matrix for large datasets. The main computational challenge of *ActiVis* is in computing the neuron activation matrix over large datasets. Here, we describe our scalable approach whose time complexity is linear in the number of data instances. We first create a matrix S (#instances $\times$ #subsets) that describes all instance-to-subset mappings. Once a model predicts labels for instances, it produces an activation matrix A (#instances $\times$ #neurons) for each variable node. By multiplying these two matrices (i.e., $S^T A$), followed by normalization, we obtain a matrix containing all subsets' average neuron activation values, which are visualized in the neuron matrix view. As the number of instances dominates, the above computation's time complexity is linear in the number of instances. In practice, this computation roughly takes the same amount of time as testing a model. We have tested *ActiVis* with many datasets (e.g., one with 5 million training instances). *ActiVis* can now scale to any data sizes that FBLearner supports (e.g., billion-scale or larger).

Implementation Details

The visualization and interactions are implemented mainly with React.js.[5] We additionally use a few D3.js V4 components.[6] The computation graph is visualized using Dagre,[7] a JavaScript library for rendering directed graphs. All the backend code is implemented in Python (including scikit-learn[8] for t-SNE) and the activation data generated from backend are passed to the interface using the JSON format.

[5] React.js, `https://facebook.github.io/react/`
[6] D3.js, `https://d3js.org/`
[7] Dagre, `https://github.com/cpettitt/dagre`
[8] scikit-learn, `http://scikit-learn.org/`

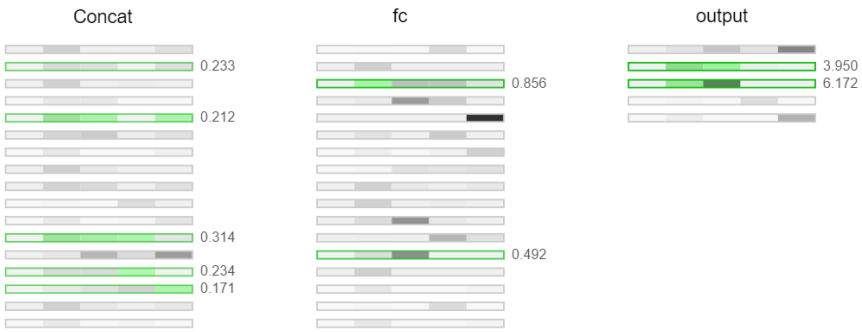


Figure 3.6: Version 1 of *ActiVis*, showing an instance's neuron activation strengths, encoded using color intensity. A main drawback of this design was that users could only see the activations for a single instance at a time. Activation comparison across multiple instances was not possible.

3.4 Informed Design through Iterations

The current design of *ActiVis* is the result of twelve months of investigation and development effort through many iterations.

Unifying instances and subsets to facilitate comparison of multiple instances. The first version of *ActiVis*, depicted in Figure 3.6, visualizes activations for all layers (each column group represents a single layer). A main drawback of this design is that users can only see the activations for a single instance at a time; they cannot compare multiple instances' activations. While, for the subsets, we use an approach similar to *ActiVis*'s design (each dot represents the average values for the subset), we encode activations for a given instance using background color (here, in green). This means that the visualization cannot support activation comparison across multiple instances. This finding prompted us to unify the treatment for instances and subsets to enable comparison across them. Figure 3.7 shows our next design iteration that implements this idea.

Separating program and data to handle complex models. Although the updated version (Figure 3.7) shows activations for multiple instances, which helps users explore more information at once, it becomes visually too overwhelming when visualizing large, com-

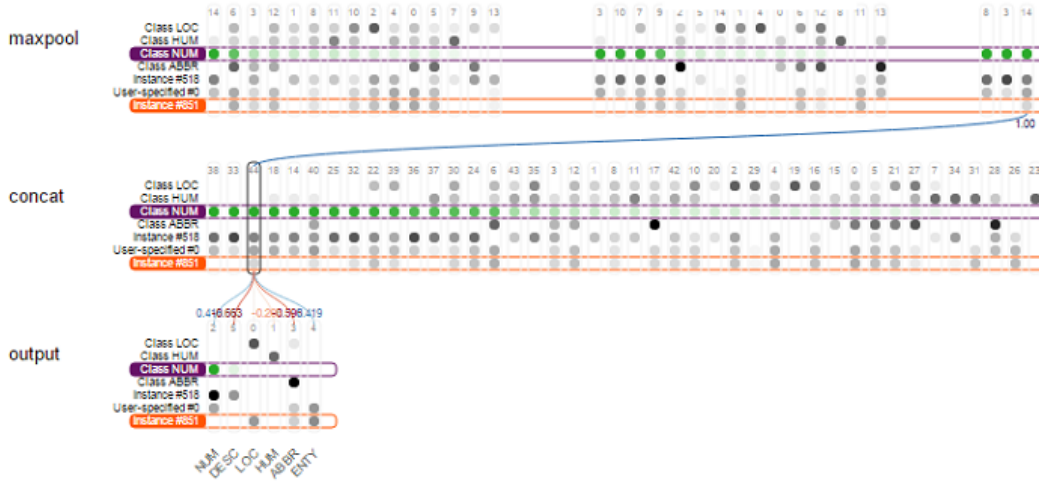


Figure 3.7: Version 2 of *ActiVis*, which unified instance- and subset-level activation visualization. This design was too visually overwhelming and did not scale to complex models, as it allocated a matrix block for each operator; a complex model could have close to a hundred operators.

plex models. Some engineers expressed concern that this design might not generalize well to different models. Also, engineers are often interested in only a few variable nodes, rather than looking at many variable nodes. Therefore, we decided to separate the visualization of the model architecture and the activations for a specific variable node.

Presenting 2-D projection of instances. One researcher suggested that *ActiVis* should provide more detail for each neuron, in addition to *average* activations. Our first solution was to present statistics (e.g., variance) and distributions for each neuron. However, some researchers cautioned that this approach could be misleading, because these summaries might not fully capture high-dimensional activation patterns. This prompted us to add the projected view (t-SNE), which enabled users to better explore the high-dimensional patterns (see Figure 3.4).

3.5 Case Studies and Usage Scenarios

To better understand how *ActiVis* may help Facebook machine learning users with their interpretation of deep neural network models, we recruited three Facebook engineers and data scientists to use the latest version of *ActiVis* to explore text classification models relevant to their work. We summarize key observations from these studies to highlight *ActiVis*'s benefits (Section 3.5.1). Then, based on observations and feedback from these users and others who participated in our earlier participatory design sessions, we present example usage scenarios for ranking models to illustrate how *ActiVis* would generalize (Section 3.5.2).

3.5.1 Case Studies: Exploring Text Classification Models with *ActiVis*

Participants and Study Protocol

We recruited three Facebook engineers and data scientists to use our tools (their names substituted for privacy):

> *Bob* is a software engineer who has expertise in natural language processing. He is experimenting with applying text classification models to some Facebook experiences, such as for detecting intents from a text snippet, like understanding when the user may want to go somewhere [3]. For example, suppose a user writes *"I need a ride"*, Bob may want the models to discover if the user needs transportation to reach the destination. He is interested in selecting the best models based on experimenting with many parameters and a few different models, as in [85, 95].
>
> *Dave* is a relatively new software engineer. Like Bob, he is also working with text classification models for user intent detection, but unlike Bob, he is more interested in preparing training datasets from large collections of databases.
>
> *Carol* is a data scientist who holds a Ph.D. in the area of natural language processing.

Unlike Bob and Dave, she is working with many different machine learning tasks, focusing on textual data.

We had a 60-minute session with each of the three participants. For the first 20 minutes, we asked them a few questions about their typical workflows, and how they train models and interpret results. Then we introduced them to *ActiVis* by describing its components. The participants used their own datasets and models, available from FBLearner Flow. After the introduction, the participants used *ActiVis* while thinking aloud. They also gave us feedback on how we could further improve *ActiVis*. We recorded audio during the entire session and video for the last part.

Key Observations

We summarize our key observations from interacting with the three participants into the following three themes, each highlighting how our tool helped them with the analysis.

Spot-checking models with user-defined instances and subsets. *ActiVis* supports flexible subset definition. This feature was developed based on the common model development pattern where practitioners often curate "test cases" that they are familiar with, and for which they know their associated labels. For example, a text snippet "Let's take a cab" should be classified as a positive class of detecting transportation-related intent. Both Bob and Dave indeed found this feature useful (i.e., they also had their own "test cases"), and they appreciated the ability to specify and use their own cases. This would help them better understand whether their models are working well, by comparing the activation patterns of their own instances with those of other instances in the positive or negative classes. Bob's usage of *ActiVis* and comments echo and support the need for subset-level visualization and exploration, currently inadequately supported by existing tools.

Graph overview as a crucial entry point to model exploration. From our early participatory design sessions, we learned that *ActiVis*'s graph overview was important for practitioners who work with complex models whose tasks only require them to focus on

specific components of the models. Bob, who works with many different variations of text classification models, has known that the model he works with mainly uses convolution operations and was curious to see how the convolution works in detail. When he launched *ActiVis*, he first examined the model architecture around the convolution operators using the computation graph panel. He appreciated that he could see how model training parameters are used in the model, which helped him develop better understanding of the internal working mechanism of the models. For example, he found how and where *padding* are used in the models by exploring the graph [26]. After he got a better sense about how the model function around the convolution operators, he examined the activation patterns of the convolution output layer. This example shows that the graph overview is important for understanding complex architectures and locating parts that are relevant to the user's tasks. In other words, the graph serves as an important entry point of Bob's analysis. Existing tools assuming user familiarity with models may not hold in real-world large-scale deployment scenarios.

Visual exploration of activation patterns for evaluating model performances and for debugging hints. One of the main components of *ActiVis* is the visual representation of activations that helps users easily recognize patterns and anomalies. As Carol interacted with the visualization, she gleaned a number of new insights, and a few hints for how to debug deep learning models in general. She interactively selected many different instances and added them to the neuron activation matrix to see how they activated neurons. She found out that the activation patterns for some instances are unexpectedly similar, even though the textual content of the instances seem very different. Also, she spotted that some neurons were not activated at all. She hypothesized that the model could be further improved by changing some of the training parameters, so she decided to modify them to improve the model. While the neuron activation panel helps Carol find models that can be further improved, Bob found some interesting patterns from the activation patterns for the convolution output layer. He quickly found out that some particular words are

highly activated while some other words, which he thought can be highly activated, do not respond much. This helped him identify words that are potentially more effective for classification. The examples above demonstrate the power of visual exploration. *ActiVis* helps users recognize patterns by interacting with instances and instance subsets they are familiar with.

3.5.2 Usage Scenario: Exploring Ranking Models

As there are many potential uses for *ActiVis* at Facebook, we also discussed with a number of researchers and engineers at different teams to understand how they may adopt *ActiVis*. Below, we present a usage scenario of *ActiVis* for exploring ranking models, based on our discussion. We note the scenario strongly resembles others that we have discussed so far; this is encouraging because enabling *ActiVis* to generalize across teams and models is one of our main goals.

Alice is a research scientist working with ranking models, one of the important machine learning tasks in industry. The ranking models can be used to recommend relevant content to users by analyzing a large number of numerical features extracted from databases [18, 74]. Alice is experimenting with deep neural network models to evaluate how these models work for a number of ranking tasks. She often performs subset-based analysis when examining model performance, such as defining subsets based on categories of page content. Subset-based analysis is essential for Alice, because she works with very large amount of training data (billions of data points, thousands of features). *ActiVis*'s instance-based exploration feature is not yet helpful for Alice, since she is still familiarizing herself with the data and has not identified instances that she would like to use for spot-checking the model. In *ActiVis*, Alice is free to use either or both of instance- and subset-based exploration. For new, unfamiliar datasets, Alice finds it much easier to start her analysis from the high level, then drill down into subsets, using attributes or features.

Alice has trained a fully-connected deep neural network model with some default pa-

rameters. When she launches *ActiVis*, she first examines the output layer to see how the activation patterns for the positive and negative classes may be different. To her surprise, they look similar. Furthermore, by inspecting the neuron activation matrix view, she realizes that many neurons are not activated at all — their activation values are close to 0. This signals that the model may be using more neurons than necessary. So, she decided to train additional models with different parameter combinations (e.g., reduce neurons) to relieve the above issue.

The performances of some models indeed improve. Happy with this improvement, Alice moves on to perform deeper analysis of the trained models. She first creates a number of instance subsets by using *features*. She utilizes 50 top features known to be important for ranking. For categorical features, she defines a subset for each category value. For numerical features, she quantizes them into a small number of subsets based on the feature value distribution. *ActiVis*'s neuron activation matrix view visualizes how the subsets that Alice has defined are activating the neurons. Maximizing the matrix view to take up the entire screen (and minimizing the computation graph view), Alice visually explores the activation matrix and identifies a number of informative, distinguishing activation patterns. For example, one neuron is highly activated for a single subset, and much less so for other subsets, suggesting that neuron's potential predictive power. With *ActiVis*, Alice can train models that perform well and understand how the models capture the structure of datasets by examining the relationships between features and neurons.

3.6 Discussion and Future Work

Visualizing gradients. Examining *gradients* is one of the effective ways to explore deep learning models [93, 45]. It is straightforward to extend *ActiVis* to visualize gradients by replacing activations with gradients. While activation represents forward data flow from input to output layers, gradient represents backward flow. Gradients would help developers

to locate neurons or datasets where the models do not perform well.

Real-time subset definition. For *ActiVis* to work with a new subset, it needs to load the dataset into RAM to check which instances satisfy the subset's conditions. Currently, it is not of high priority for the above process to be performed in real time, because users often have pre-determined subsets to explore. We plan to integrate dynamic filtering and searching capabilities, to speed up both subset definition and instance selection.

Automatic discovery of interesting subsets. With *ActiVis*, users can flexibly specify subsets in infinitely many ways. One of the engineers commented that *ActiVis* could help suggest interesting subsets for exploration, based on heuristics or measures. For example, for text datasets, such a subset could include phrases whose activation patterns are very similar or different to those for a given instance or class.

Supporting input-dependent models. An interesting research direction is to extend *ActiVis* to support models that contain variable nodes whose number of neurons changes depending on the input (e.g., the number of words in a document), and to study the relationships between neurons and subsets for such cases.

Understanding how *ActiVis* informs model training. We plan to conduct a longitudinal study to better understand *ActiVis*'s impact on Facebook's machine learning workflows, such as how *ActiVis* may inform the model training process. For example, a sparse neuron matrix may indicate that a model is using more neurons than needed, which could inform engineers on their decisions for hyperparameter tuning.

3.7 Conclusion

We presented *ActiVis*, a visual analytics system for deep neural network models. We conducted participatory design session with over 15 researchers and engineers across many teams at Facebook to identify key design challenges, and based on them, we distilled three main design goals: (1) unifying instance- and subset-level exploration; (2) tight integration

of model architecture and localized activation inspection; and (3) scaling to industry-scale data and models. *ActiVis* has been deployed on Facebook's machine learning platform. We presented case studies with Facebook engineers and data scientists, and usage scenarios of how *ActiVis* may be used with different applications.

PART II

DATA-DRIVEN MODEL AUDITING

Overview

While tools like *ActiVis* promote people's understanding of a model by visualizing how it responds to data instances and subsets, the interpretation of a model is only one part of many different tasks in applied machine learning. Building machine learning models involves many different analytics tasks (e.g., feature extraction, model selection) and often requires analysis of how input datasets affect results over a long machine learning pipeline. Thus, it is important to assist researchers and practitioners who work on various stages of a machine learning workflow, to identify important, and potentially problematic data groups, so that they can discover insights and potentially fix the problems. In particular, this part describes two work in this line of research:

- *MLCube* (Chapter 4) on supporting model comparison using user-specified data subsets and data cube analysis;
- *FairVis* (Chapter 5) on discovering intersectional bias in machine learning models with the help of automated techniques.

CHAPTER 4

MLCUBE: INTERACTIVE MODEL COMPARISON WITH DATA CUBE ANALYSIS

In Part I, we presented how the subset-level analysis can be used for interpreting machine learning models, however, one challenge is a subset can be specified in many different ways. This chapter presents *MLCube*, a data cube inspired framework that enables users to define instance subsets using feature conditions and computes aggregate statistics and evaluation metrics over the subsets. We also design *MLCube Explorer*, an interactive visualization tool for comparing models' performances over the subsets. Users can interactively specify operations, such as drilling down to specific instance subsets, to perform more in-depth exploration. Through a usage scenario, we demonstrate how *MLCube Explorer* works with a public advertisement click log data set, to help a user build new advertisement click prediction models that advance over an existing model.

4.1 Introduction

As machine learning systems become more widely adopted, they are becoming increasingly complex. Applying machine learning techniques on large-scale, real-world problems often entails many steps, including feature extraction, feature transformations, model selection, and model evaluation [21, 157]. Each component itself may introduce its own complexity.

This chapter is adapted from work appeared at HILDA 2016 [88].

For example, it is non-trivial to extract meaningful feature sets from a large number of attributes [10]. In practice, as machine learning systems increase in size and complexity, they are often viewed as "black boxes," as there are no effective ways for understanding the internal mechanisms of these complex systems or interpreting their model results [100, 149].

The importance of helping users interpret machine learning models has received increasing attention. Recent work [102, 9, 100, 149] highlighted that while overall model accuracy can be used to select models, users often want to understand why and when a model would perform better than others, so that they can trust the model and know how to improve the model. Current interpretation approaches often focus on explaining single models (e.g., computing feature importance from a boosted tree), but they cannot be directly applied on other models (e.g., neural networks) since the internal working mechanisms of different models can vary widely [100, 149]. Current visualization approaches primarily support instance-based explanation (e.g., how individual instances contribute to a model's accuracy) [9, 141]; more work is needed to find out if they may scale up to larger data sets or more complex systems.

Introducing *MLCube Explorer*. We present an interactive visualization tool for comparing machine learning models' performances and exploring model results using data cube analysis. Our goal is to help users interactively explore and determine the right abstraction level of analysis — as comparing two models by their overall accuracies is often too coarse and not conducive to discovering contributing causes; and inspecting individual instances within a large data set is too fine-grained and may not scale — our work helps user reach the "happy medium."

Specifically, our proposed *MLCube* framework enables users to define **instance subsets** using *relational selections* over features, and compute aggregate statistics and evaluation metrics over the subsets. Through our *MLCube Explorer* (Figure 4.1), users can visually explore these subsets and interactively specify operators to further analyze the results. For

Figure 4.1: A screenshot of *MLCube Explorer*, our interactive visualization tool for analyzing and comparing machine learning results. Each row represents a subset. The *subset summary view* (middle) visually shows several statistics for each subset (e.g., count, proportion of positive instances, distribution of prediction scores, and model's accuracy). The *correlation matrix view* (right) visualizes accuracy differences between two models for a subset combination (e.g., `user_age_group=1 AND position=3`). A cell with a larger circle means there are more instances in that subset combination. Yellow means Model A outperforming Model B; green means Model B outperforming Model A. The darker the color, the greater the performance difference. Users can interact with the interface in several ways, including drill-down, sorting subsets, and adding new subsets.

example, they can drill down into subsets to explore relationships among features and examine how they affect model results. Users can freely define subsets with both raw data attributes and transformed features.

Drilling down model results. *MLCube* introduces a new way for users to select instance subsets using both data *attributes* (e.g., the titles of text documents) and *features*,

which are often derived from attributes (e.g., number of terms in the titles). Prior research has shown that leveraging *features* in explanations is a key to interpreting machine learning results [102, 27, 100]. The feature-based analysis can generalize to any models that share the same feature sets, unlike model-specific explanations [173]. Our approach advances over prior work [38], by allowing users to interactively select subsets based on their knowledge of any feature transformations that have been carried out, and also keep track of the intermediate stages in the workflow. This functionality is important because raw data attributes are often transformed into features through feature engineering (e.g., as in calculating the number of terms from the titles) [10]. Slicing results by features may impede user understanding, since revealing relationships between the data attributes and the behavior of machine learning algorithms could accelerate understanding of model behavior [126, 141].

Interactive visualization. To help users quickly get an overview of the data and model results and spot interesting patterns and anomalies, *MLCube Explorer* allows users to visually explore aggregate statistics over subsets of data instances and interactively drill down into models. This enables users to find interesting patterns between features and model results, leading to discovering insights that help them understand the mechanisms of the models and further improve their performance.

Our contributions are:

- *MLCube*, a data cube inspired framework that enables users to explore aggregate statistics and evaluation metrics over the user-defined subsets (Section 4.3);
- *MLCube Explorer*, a visualization tool for interactively exploring *MLCube* for analyzing and comparing models' performances (Section 4.4).

We demonstrate how *MLCube Explorer* works with a public advertisement click log data set through a usage scenario of building advertisement click prediction models (Section 4.5).

4.2 Background: A Typical Machine Learning Pipeline

In this section, we describe a typical workflow of building machine learning models for data sets (see Figure 4.2), to motivate and provide the context for *MLCube*'s contributions. For example, as *MLCube* is defined over data, features, and model results, we will first explain the terminology and symbols for describing them. Building machine learning models in practice often involves several steps, including data pre-processing, feature extraction, feature transformations, model building, and model evaluations [21, 157]. To illustrate with a concrete example, we use the task from the 2012 KDD Cup competition (Track 2),[1] whose goal is to build advertisement click prediction models.

A **raw data table** $\mathbf{R}$ is a relation having a set of attributes A and consisting of a set of instances. Each instance $\mathbf{r}_i \in \mathbf{R}$ consists of a set of attribute values $\mathbf{r}_i[A_j]$. An attribute value could be either single-valued or multi-valued, where its data type could be integer, float, or text. For the case of advertisement prediction, each instance represents an event in which an advertisement is shown to a user under a certain setting. Its attributes include user ID, age, gender, ad ID, the title of an ad, query ID, query text, position of ad on a webpage, binary value of whether the user clicked the ad (1 if clicked; 0 otherwise), etc. Using the database terminology, the raw data table $\mathbf{R}$ can be thought of as a joined relation of a log table (i.e., fact table) with several entity tables (e.g., Users, Ads) [105].

The next step is the *feature extraction* or *feature transformation* procedure that constructs a set of features from a raw data table $\mathbf{R}$. This step consists of a set of **feature functions** $\mathcal{F}$, taking a raw data table $\mathbf{R}$ as an input and producing a **feature vector table** $\mathbf{X}$ (and **labels** $\mathbf{y}$) that will be used as an input for a learning model [11]. Each feature function $F_j \in \mathcal{F}$ produces a j-th feature value x_{ij} for a given instance r_i. In other words, each instance r_i will be transformed into a feature vector $\mathbf{x}_i = (x_{i1}, ..., x_{ij}, ...)$ and a label $y_i \in \{0, 1\}$. Some feature functions may simply select an attribute (e.g., user's age), while

[1] `http://www.kddcup2012.org/c/kddcup2012-track2`

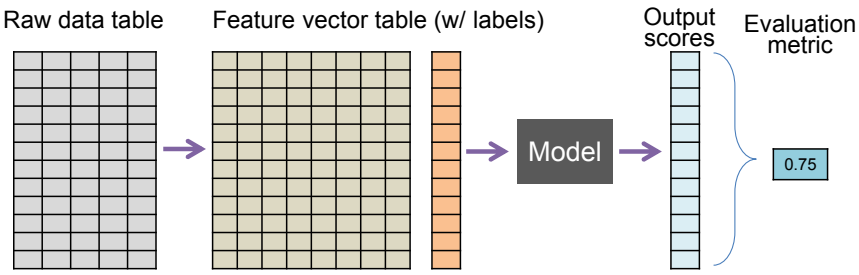


Figure 4.2: Typical machine learning pipeline from raw datasets to metric scores for evaluation.

others may perform computation. For example, an *average function* may compute the average click-through rate for each ad ID; a *tf-idf function* may calculate tf-idf text similarity between a query and the title of an advertisement [184].

Given a feature vector table and labels, engineers would then run different machine learning algorithms on them. Once a **prediction model** $h(\mathbf{X}, \mathbf{y})$ is constructed (e.g., logistic regression), it will be used to classify test instances $\mathbf{x}_i$. For each instance, the model produces a **prediction score** s_i and determines its corresponding predicted label $\hat{y}_i \in \{0, 1\}$. The performance of the models is evaluated using a **evaluation measure** l (e.g., accuracy, AUC score), which takes as input a list of (label y, score s or predicted label $\hat{y}$) pairs, and outputs a single value (i.e., the measure).

4.3 *MLCube*: Data Cubes for Machine Learning

We present *MLCube*, a data cube inspired framework for analyzing machine learning model results. Our approach enables users to flexibly analyze and understand model results at the *subset* level, through interactively exploring and generating a wide range of instance subsets (see Figure 4.3). A subset is defined as a relational selection over a feature vector table $\mathbf{X}$ or the raw data table $\mathbf{R}$ (e.g.,`user_gender = 'female'`). *MLCube* computes aggregate statistics (e.g., accuracy) for all user-defined subsets.

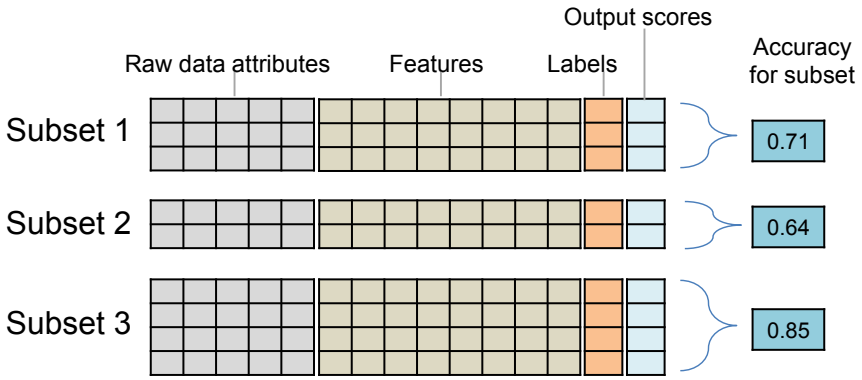


Figure 4.3: *MLCube* enables subset-level analysis of machine learning results by computing aggregate statistics (e.g., accuracy) for a subset of instances.

While *online analytical processing (OLAP)* is traditionally defined over a fact table consisting of a set of dimension attributes and a measure attribute, *MLCube* is defined over $\mathbf{R} \bowtie \mathbf{X} \bowtie \mathbf{y} \bowtie \mathbf{s} \bowtie \hat{\mathbf{y}}$. The primary keys (PKs) of all five relations are instances' unique ID, and all join conditions apply only on the PKs. As all intermediate data are included, a subset can be defined not only over features, but over data attributes (e.g., `ad_title contains 'car'`), or over a combination of multiple components (e.g., `ad_title contains 'car' AND tfidf_sim_query_title >= 0.7 AND label = 1`).

For the *measure attributes* of the cube (i.e., values to be aggregated), we find the following measures particularly useful for analyzing model results:

- **Accuracy (or any evaluation measure, such as AUC) for a model:** Computing accuracy by subsets allows engineers to understand which data regions work better or worse for a selected model. If accuracy for a certain subset is relatively low, engineers may want to inspect the corresponding instances.

- **Accuracy difference between two models:** Comparing a new model to a control model in a subset-level is particularly useful for model selection, as it could describe which parts of data helps improve or degrade the model performance.

- **Number of instances** *(model-independent)*: Instance counts help engineers understand the data distributions and find important subsets (e.g., they may ignore subsets with very small number of instances).

- **Proportion of positive instances** *(model-independent)*: Helps with feature engineering by showing which features are more discriminative.

In addition to the above measures, there could be many useful measures we can define. For example, computing score distributions of positive (or negative) instances helps people understand the result of a model.

While subsets can be defined as any relational selection with SQL-like expression, a set of dimension attributes (i.e., categorical) is often selected in practice because of scalability issues, since infinite number of subsets could be generated. By default, *MLCube* selects all categorical attributes (i.e., cardinality less than certain threshold) and create discrete bins for selected numerical (continuous) attributes and features. To speed up statistics computation over subsets, the *MLCube* is then partially materialized for these subsets. In our implementation, we use an algorithm described in [132] (Algorithm 1) with Apache Spark[2] [190] and constrain the maximum number of dimensions to 4.

4.4 Visual Exploration of *MLCube*

This section presents *MLCube Explorer*, an interactive visualization tool for exploring machine learning results using *MLCube*. Interactive visualizations has been proven to be very effective for finding interesting patterns and spotting anomalies from large, multidimensional data by effectively representing data and allowing users to interact with them [164, 165, 9, 120]. We introduce the interface of our visualization tool and describe operations for users to interact with the interface.

[2]Apache Spark, `http://spark.apache.org/`

4.4.1 User Interface

Figure 4.1 shows a screenshot of *MLCube Explorer*, visualizing the performances of (one or) two user-selected models by subsets. Each row represents a subset. By default, we show all subsets consisting of one selection predicate chosen from feature vectors. The column of each subset row is divided into two areas: (1) the **subset summary view** which shows summary statistics for each subset and (2) the **correlation matrix view** which visualizes its pairwise correlations to other subsets. As for the subset summary view, we visualize the number of instances, the proportion of positive instances, the prediction score distributions of positive/negative instances for each model, and the accuracy value for each model. As for the correlation matrix view, each cell visualizes the accuracy difference between two models for a subset combination (e.g., `user_age_group=1 AND position=3`). A cell with a larger circle means there are more instances in that subset combination. Yellow means Model A outperforming Model B; green means Model B outperforming Model A. The darker the color, the greater the performance difference.

4.4.2 Interactive Operations

Users can interact with the interface to further explore *MLCube* using the following operations.

1. **Drill-down/Roll-up into a subset:** By clicking a subset (e.g., `user_age_group = 0`), its predicate will be applied to all other subsets, updating all values and visual elements in all rows and cells.

2. **Adding user-defined subsets:** Define a new subset based on a user-defined relational selection predicate. The new subset will be added as a new row.

3. **Using different measures:** Different measures may be used in the correlation matrix view (e.g., AUC score difference, proportion of positive instances).

4. **Sorting subsets:** Subset rows can be sorted using any measure attributes (e.g., count) to help reveal interesting patterns and spot anomalies.

4.4.3 System Implementation

We implemented (1) a simple declarative machine learning framework following the pipeline in Section 4.2, (2) *MLCube* which works on top of the framework (Section 4.3), and (3) *MLCube Explorer* (Section 4.4). The framework is implemented based on the pipeline introduced in Section 4.2 using Python, scikit-learn, and PostgreSQL. Within the framework, we implemented several learning features presented in the report by the winner of the KDD Cup [184] and implemented several models, including logistic regression, decision tree, and boosted tree, also based on the report. As we mentioned earlier, *MLCube* is partially materialized with a algorithm described in [132]. *MLCube Explorer* is written in HTML, JavaScript, and D3.js. It can run on any modern web browser. When a user specifies two created models to compare, the server returns the corresponding *MLCube* in JSON format and the client code generates the visualization.

4.5 Usage Scenario

This section presents a usage scenario for *MLCube Explorer* to demonstrate how it may help our user Jane, a machine learning engineer working at a search engine company, to build new advertisement click prediction models that advance over an existing model.

Jane uses a public data set from the 2012 KDD Cup competition.[3] It is an advertisement click log from the Tencent search engine, soso.com. Each data instance describes information about a user, an ad, a query, and whether the user clicked the ad. Jane implements some of the learning features presented in the winning team's report [184] and created a

[3] `http://www.kddcup2012.org/c/kddcup2012-track2`

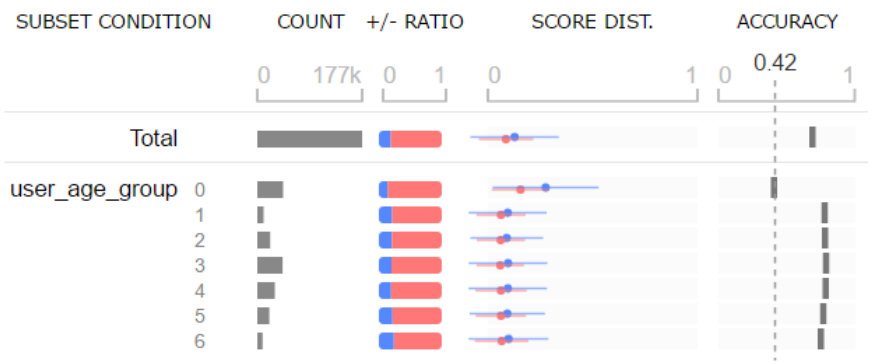


Figure 4.4: Our user Jane finds that a subset of instances "`user_age_group = 0`" performs distinctly worse than the other age groups, indicated by the left-most solid bar in the accuracy column.

few models, including logistic regression, decision tree, and boosted tree.

Recognizing data encoding issue. Jane begins her exploration by visualizing the existing model to understand its performance. She quickly finds that a subset of instances "`user_age_group = 0`" performs distinctly worse than the other age groups, indicated by the left-most solid bar in the accuracy column in Figure 6.13. To understand why, she examines how the age groups were defined. She realizes that the feature function that generates this feature has encoded null data as 0 and considered this feature as numerical variables. She thinks that this might cause the degraded performance. To fix this issue, she redefines this feature as a categorical variable, instead of a numerical variable, which could improve the performance of the model by separating the null instances from others.

Analyzing the performance improvement. After getting the hints for improving the model performance, she now would like to try different learning algorithms and compare their performance with that of the baseline model. To create a visualization, she sets the baseline boosted tree model as model A (shown in dark yellow in Figure 4.5) and picks a logistic regression model with additional features as model B (shown in green). The visualization shows that, overall, model B outperforms model A. In particular, Jane sees model B has significantly improved over model A for the subset "`user_age_group =`

Figure 4.5: Example of analyzing performance improvement. Jane sees model B has significantly improved over model A for the subset "`user_age_group = 0`". She drills down into that subset by clicking it and observes interesting patterns between accuracy and the `tfidf_sim_query_title` feature.

`0`" (Figure 4.5a). To further analyze this subset, she drills down into it by clicking it, and she observes a few important patterns for the `tfidf_sim_query_title` feature[4] (see Figure 4.5b): (1) the majority of model B's improvement over model A comes from subsets with lower similarity scores (the wide gaps between yellow and green bars) — this means model B is quite accurate even when the advertisement titles are not that similar to the user's search query; (2) the accuracy difference between model A and B decreases as the

[4]This feature measures similarity between a query and the title of an ad by representing each text field as a TF-IDF term vector and computing cosine similarity between two vectors [184].

similarity increases. In addition to the "`user_age_group = 0`" subset, she finds out that model B outperforms model A for several other subsets. Thanks to these discoveries, she decides to look into the model further by creating more variations with different parameters and features.

4.6 Future Work

This work opens up many interesting future research challenges. First, efficient materialization techniques can be integrated (e.g., by using monotonicity property, parallel computation) to speed up the exploration of data cube [132]. As the cube is accessed by interactive tools, it would also be possible to interactively materialize cubes while users navigate cubes by predicting the next possible user steps as in [91, 120]. In addition, efficient techniques for ranking interesting subsets (e.g., subsets with the largest accuracy differences between models) can help users explore a very large number of subsets [49, 154, 84]. Finally, conducting user studies can help evaluate how our tool can help machine learning engineers ease their workflow of developing effective machine learning models with a deeper understanding of the relationships between data and models.

CHAPTER 5

FAIRVIS: DISCOVERING INTERSECTIONAL BIAS IN MACHINE LEARNING

One of the greatest use cases of interactive methods for exploring many instance subsets over a machine learning pipeline (like *MLCube* in the previous chapter) is fairness auditing. Despite the benefits machine learning systems may bring, models can reflect, inject, or exacerbate implicit and explicit societal biases into their outputs, disadvantaging certain demographic subgroups. Discovering which biases a machine learning model has introduced is a great challenge, due to the numerous definitions of fairness and the large number of potentially impacted subgroups. This chapter presents *FairVis*, a mixed-initiative visual analytics system that integrates a novel subgroup discovery technique for users to audit the fairness of machine learning models. Through *FairVis*, users can apply domain knowledge to generate and investigate known subgroups, and explore suggested and similar subgroups. *FairVis* demonstrates how interactive visualization may help data scientists and the general public understand and create more equitable algorithmic systems.

5.1 Introduction

In recent years, significant strides have been made in machine learning, enabling automated, data-driven systems to tackle ever more challenging and complex tasks. Many of

This chapter is adapted from work appeared at IEEE VAST 2019 [30].

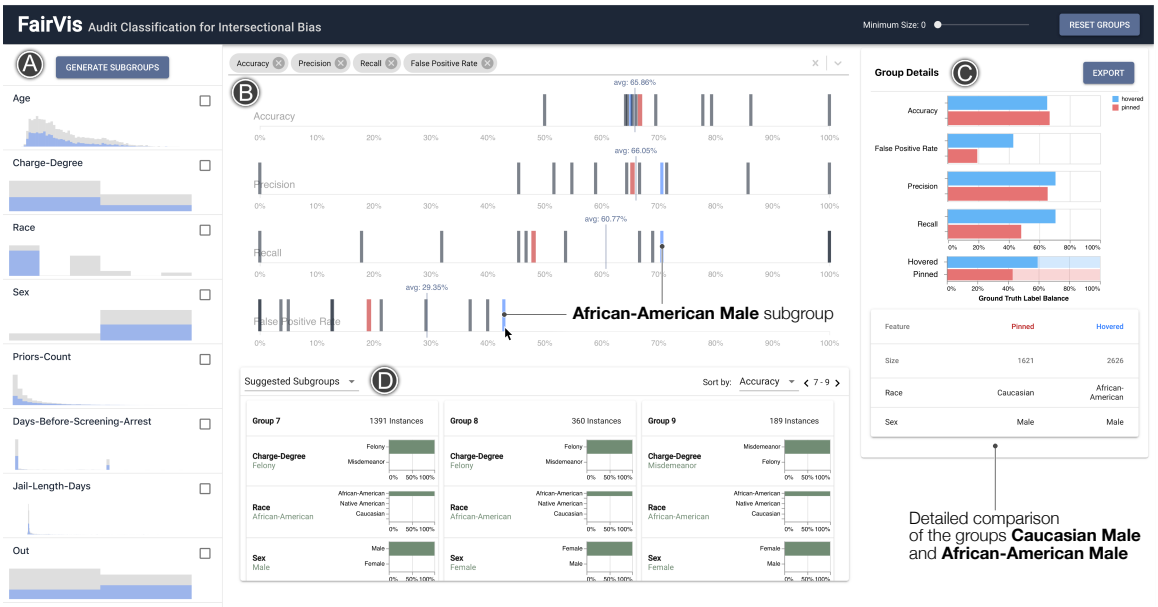


Figure 5.1: *FairVis* integrates multiple coordinated views for discovering intersectional bias. Above, our user investigates the intersectional subgroups of *sex* and *race*. **A.** The *Feature Distribution View* allows users to visualize each feature's distribution and generate subgroups. **B.** The *Subgroup Overview* lets users select various fairness metrics to see the global average per metric and compare subgroups to one another, e.g., pinned **Caucasian Males** versus hovered **African-American Males**. The plots for *Recall* and *False Positive Rate* show that for African-American Males, the model has relatively high recall but also the highest false positive rate out of all subgroups of sex and race. **C.** The *Detailed Comparison View* lets users compare the details of two groups and investigate their class balances. Since the difference in False Positive Rates between Caucasian Males and African-American Males is far larger than their difference in base rates, a user suspects this part of the model merits further inquiry. **D.** The *Suggested and Similar Subgroup View* shows suggested subgroups ranked by the worst performance in a given metric.

the new domains in which these novel techniques are being applied are human-focused and consequential, including hiring, predictive policing, predicting criminal recidivism, and pedestrian detection. The latter two cases are examples where differing levels of predictive accuracy have been observed for different demographic groups [182, 44].

When deploying machine learning to these societally impactful domains, it is vital to understand how models are performing on all different types of people and populations. Machine learning algorithms are usually trained to maximize the overall accuracy and performance of their model, but often do not take into account disparities in performance

between populations. The trained models thus provide no guarantees as to how well they will perform on different subgroups of a dataset.

The potential disparity in performance between populations may have many sources; a machine learning model can naturally encode implicit and explicit societal biases [22], which is often referred to as algorithmic bias. Performance disparity can arise for a variety of reasons: the training data may not be representative, either in terms of its representation of different demographic groups or within a particular demographic group; the training data labels may have errors which reflect societal biases, or be an imperfect proxy for the ultimate learning task; unequal rates of labels across demographic groups; the model class may be overly simple to capture more nuanced relationships between features for certain groups; and more [44]. A stark example of algorithmic bias in deployed systems was discovered by Buolamwini and Gebru's Gender Shades study [28], who showed that many commercially available gender classification systems from facial image data had accuracy gaps of over 30% between darker skinned women and lighter skinned men. While the overall models' accuracies hovered around 90%, darker skinned women were classified with accuracy as low as 65% while the models' accuracies on lighter skinned men were nearly 100%.

In order to discover and address potential issues before machine learning systems are deployed, it is vital to audit machine learning models for algorithmic bias. Unfortunately, discovering biases can be a daunting task, often due to the inherent intersectionality of bias as shown by Buolamwini and Gebru [28]. *Intersectional bias* is bias that is present when looking at populations that are defined by multiple features, for example "Black Females" instead of just people who are "Black" or "Female". The difficulty in finding intersectional bias is pronounced in the Gender Shades study introduced above — while there were performance differences when looking at sex and skin color individually, the significant gaps in performance were only found when looking at the intersection of the two features. An example of how aggregated measures can hide intersectional bias can be

seen in Figure 5.2.

In addition to the intersectional nature of bias, addressing bias is challenging due to the numerous proposed definitions of unfairness. The metrics for measuring a model's fairness include measuring a model's group-specific false positive rates, calibration, and more. While a user may decide on one or more metrics to focus on, achieving true algorithmic fairness can be an insurmountable challenge. In Section 5.2, we describe how recent research has shown that it is often impossible to fulfill multiple definitions of fairness at once.

While it can be straightforward to audit for intersectional bias when looking at a small number of features and a single fairness definition, it becomes much more challenging with a large number of potential groups and multiple metrics. When investigating intersectional bias of more than a few features, the number of populations grows combinatorially and quickly becomes unmanageable. Data scientists often have to balance the tradeoffs between various fairness metrics when making changes to their models.

To help data scientists better audit their models for intersectional bias, we introduce ***FairVis***, a novel visual analytics system dedicated to helping audit the fairness of machine learning models. *FairVis*'s major contributions include:

- **Visual analytics system for discovering intersectional bias.** *FairVis* is a mixed-initiative system that allows users to explore both suggested and user-specified subgroups that incorporate a user's existing domain knowledge. Users can visualize how these groups rank on various common fairness and performance metrics and contextualize subgroup performance in terms of other groups and overall performance. Additionally, users can compare the feature distributions of groups to make hypotheses about why their performance differs. Lastly, users can explore similar subgroups to compare metrics and feature values.

- **Novel subgroup generation technique.** In order to aid users in exploring a combinatorially large number of subgroups, we introduce a new subgroup generation

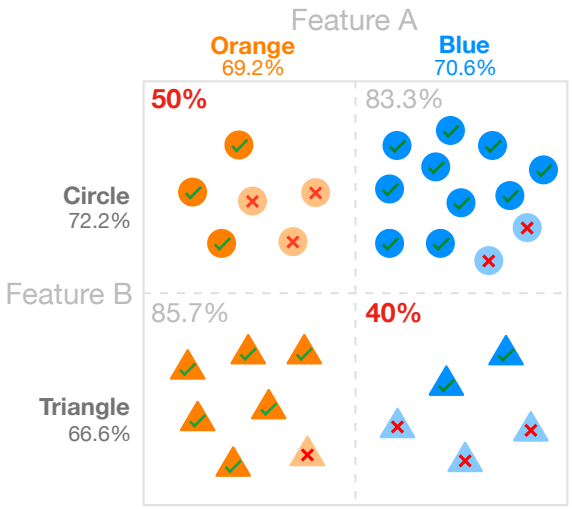


Figure 5.2: This illustrative example highlights how inequities in populations can be masked by aggregate metrics. While the classifier in this example has an accuracy of between 66.6% and 72.2% when looking at groups defined by a single feature, the accuracy drops to as low as 40% when looking at the intersectional subgroups.

technique to recommend intersectional groups on which a model may be underperforming. We first run clustering on the training dataset to find statistically similar subgroups of instances. Next, we use an entropy technique to find important features and calculate fairness metrics for the clusters. Lastly, we present users with the generated subgroups sorted by important and anomalously low fairness metrics. These automated suggestions can aid users in discovering subgroups on which a model is underperforming.

- **Method for similar subgroup discovery.** Once a subgroup for which a model has poor performance has been identified, it can be useful to look at similar subgroups to compare their values and performance. We use similarity in the form of statistical divergence between feature distributions to find subgroups that are statistically similar. Users can then compare similar groups to discover which value differences impact performance or to form more general subgroups of fewer features.

5.2 Background in Machine Learning Fairness

Significant discoveries and advances have been made in algorithmic bias detection, mitigation, and machine learning fairness in recent years. Most of the work stems from theoretical computer scientists and sociologists focusing on the mathematical foundations and societal impacts of machine learning.

A major difficulty in machine learning fairness is that it is mathematically impossible to fulfill all definitions of fairness simultaneously when populations have different base rates. This incompatibility between fairness metrics was formalized by the *impossibility theorem* for fair machine learning. Two papers [96, 61] simultaneously proved that if groups have different base rates in their labels, it is statistically impossible to ensure fairness across three base fairness metrics — balance for the positive class, balance for the negative class, and calibration of the model. Data scientists must therefore decide which fairness metrics to prioritize in a model and how to make trade-offs between metric performance.

The implications of this discovery were made apparent in the recidivism prediction tool COMPAS, a system that is used to predict the risk of letting someone go on bail. A ProPublica article [14] showed that COMPAS is more likely to rank a Black defendant as higher risk than a White defendant given that they have equal base rates. A follow-up study showed that while COMPAS is not balanced for the positive class prediction, it is well calibrated, meaning that the model provides similarly accurate scores for both groups relative to their base rates [50]. Due to inherent base rate differences, it is not possible for COMPAS to meet the all three fairness definitions at once. We explore this dataset more in Section 5.5.1.

There have been various solutions proposed for addressing algorithmic bias in machine learning across the entire model training pipeline. These range from techniques for obfuscating sensitive variables in training data [186], to new regularization parameters for training [24] and post-processing outcomes by adding noise to predictions [71]. While

these can help balance certain inequities, the impossibility theorem dictates that hard decisions will still have to be made about which fairness metrics are the most important for each problem. Ideally, over time these will become standard processes for ensuring model fairness, and tools like *FairVis* can be used to ensure their effectiveness and investigate tradeoffs between metrics.

Furthermore, important innovations have come from the machine learning community in relation to *intersectional bias*. Kearns et al. [94] proposes a framework for auditing a (possibly very large) number of subgroups for unfair treatment. Their work has the same high-level concerns that motivate this project: that there may be a very large number of intersectional groups over which one wants to satisfy some notion of fairness. However, for their work, they assume the collection of these groups is predefined for the task at hand, and construct an algorithm for creating a distribution over classifiers which (approximately) minimizes a particular fairness metric over all the subgroups simultaneously. Our work differs from theirs in several key ways. First, we aim to operate in a space where a predefined notion of groups is not necessarily available, and so cooperation between an automated system and a domain expert might be necessary to uncover subgroups whose treatment by a particular model is problematic. Second, our goal is to help a user explore their model and dataset for a deeper understanding of *why* the model might be treating particular groups very differently, a far different task compared to aiming to satisfy a particular fairness metric without delving into the data-dependent sources of this different treatment. This deeper model understanding will facilitate task-specific interventions and promote a deeper understanding of a learning task, a dataset's suitability to this task, and whether a model (class) matches the dataset and task.

5.3 Design Challenges and Goals

Our goal is to build an interactive visual interface to help users explore the fairness of their machine learning models and discover potential biases. Many of the challenges present in auditing for bias derive from the combinatorial number of subgroups generated when looking at various features. Additionally, any visual system must convey multiple fairness metrics for a subgroup. A successful visual system should allow users to narrow the large search space of possible subgroups. We formalize these important factors in the design of *FairVis* with the following key design challenges:

5.3.1 Design Challenges

C1. **Auditing the performance of known subgroups.** For many datasets and problem definitions, users already know of certain populations for which they want to ensure fair outcomes [174]. It is often cumbersome and slow to manually generate and calculate various performance metrics for subgroups. A system should enable users to generate any type of subgroup they want to investigate, and efficiently generate and calculate metrics for it [77].

C2. **Contextualizing subgroup performance in relation to multiple metrics and other groups.** To measure the severity of bias against a certain subgroup, it is important to know how the subgroup is performing in relation to the overall model. Any visual encoding of subgroup performance should convey how groups perform for different performance metrics [71] and in relation to other subgroups. Our interface should also allow users to drill down into subgroup details while maintaining the high-level view.

C3. **Discovering significant subgroups in a large search space.** When investigating intersectional bias, there could be hundreds or thousands of subgroups a user may

need to look at [94]. It is often not feasible to analyze every group, so deciding how to prioritize subgroups is an important and difficult task. Methods for discovering and suggesting potential groups can aid users in searching this large space and finding potential issues more efficiently.

C4. **Finding similar subgroups to investigate feature importance and more general groups.** When a biased subgroup has been identified, it can be informative to look at the performance of similar subgroups to draw conclusions about feature importance or to create more general groups [57, 191]. This is a difficult task since an immense number of potential subgroups have to be searched to find similar subgroups, and it is not clear how similarity between subgroups should be defined or calculated.

C5. **Emphasizing the inherent trade-offs between fairness metrics.** Classifiers are often not able to fulfill all measures of fairness if the base rates between populations are different, as proven by the *impossibility of fairness* theorem (Section 5.2). This means users often have to keep in mind the tradeoffs between fairness metrics when deciding what modifications to make to their models. It is essential to show the various fairness metrics when displaying subgroup performance and emphasize their tradeoffs.

C6. **Suggesting potential causes of biased behavior.** How to address bias in machine learning models is a difficult and open question, but there are indicators that can help users start to improve their models. Emphasizing information like ground-truth label balance, subgroup entropy, and data distribution can point users in the right direction for addressing biases [71, 99].

5.3.2 Design Goals

Using the design challenges we identified for machine learning bias discovery, we iterated and developed design goals for *FairVis*. The following goals address the challenges

presented in Section 5.3.1, and align with the primary interface components of our system:

G1. **Fast generation of user-specified subgroups.** Since users often have domain knowledge about important subgroups they want to ensure fairness for (C1), quickly generating these groups to enable investigation is vital. Users should be able to select either entire features (e.g. "race") or specific values (e.g. "white" or "black") to generate groups of any feature combination (C3). Users should then be able to explore the performance of these groups in detail.

G2. **Combined overview relationships with detailed information of subgroup performance.** To understand the magnitude and type of bias a model has encoded for a subgroup, it is important to show the performance of the group in relation to the overall and other subgroups' performance (C2). At the same time, the interface should also display detailed information about the performance of the selected subgroup (C6). We aim to achieve this by using multiple, coordinated views that can handle different fairness metrics (C5).

G3. **Suggested under-performing subgroups for user investigation.** When more than a couple of features are used to define subgroups, the number of generated groups grows combinatorially (C3). We aim to develop both an algorithmic technique for automatically discovering potentially under-performing subgroups and an intuitive visual encoding for suggesting discovered groups to the user. By suggesting these groups automatically, we can make the subgroup discovery process quicker and potentially discover groups the user had not originally thought about (C2).

G4. **Efficient calculation of similar subgroups.** For any given subgroup, there is a combinatorially large space of groups that need to be searched to find similar groups (C3). Since it is often useful to look at similar subgroups to analyze the importance of certain features or to generate more general groups, we aim to develop a technique that efficiently discover these similar groups (C4, C6).

G5. **Effective visual interfaces for subgroup comparison.** Users may want to analyze two subgroups side by side to compare their values or performance (C2). We aim to provide an intuitive interface for highlighting the differences between two groups. Users can compare these groups to help pinpoint which features or values are causing the difference in fairness metrics (C6).

5.4 *FairVis*: Discovering Intersectional Bias

From the design goals in Section 5.3.2, we have developed *FairVis*, a visual analytics system for discovering intersectional bias in machine learning models. To meet the listed design goals, we developed two novel techniques to generate underperforming subgroups and find similar subgroups. We combine these techniques in a web-based system that tightly integrates multiple, coordinated views to help users discover fairness issues in known and unknown subgroups.

Our interface consists of four primary views, the *Feature Distribution View* (Section 5.4.1), *Subgroup Overview* (Section 5.4.2), *Suggested and Similar Subgroup View* (Section 5.4.3, Section 5.4.4), and *Detailed Comparison View* (Section 5.4.5). The *Feature Distribution View* gives users an overview of the dataset distribution and allows them to generate groups to visualize in the *Subgroup Overview* Users can then add additional subgroups provided by the *Suggested and Similar Subgroup View* and compare and further analyze them in the *Detailed Comparison View* Each section of our interface aligns with one of the stated design goals, addressing each desired feature.

5.4.1 Feature Distribution View & Subgroup Creation

The left sidebar, or *Feature Distribution View*, acts as both a high-level overview of a dataset's distribution and the interface for generating user-specified subgroups. As a starting place for *FairVis*, the *Feature Distribution View* helps users develop an idea of their

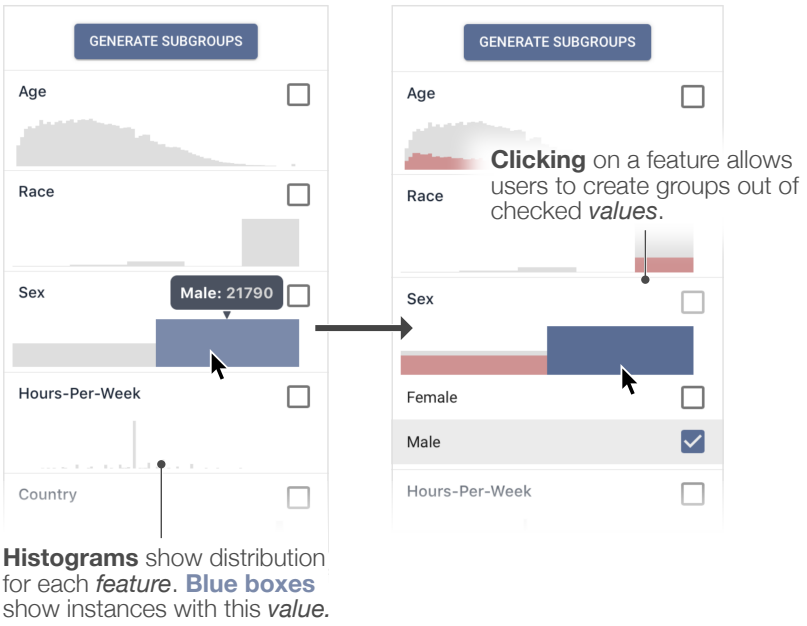


Figure 5.3: The *Feature Distribution View* allows users to explore both the distributions of each feature in the entire dataset and also create user-specified groups out of features or specific values. When a user hovers over a bar such as "Male", it shows the number of instances for that value. Red bars show the distribution of the pinned group (in this case "White Males") from the *Subgroup Overview* .

dataset's makeup and begin auditing subgroups right away.

Feature distribution. A large part of understanding model performance is understanding how the data used to train a model is distributed (C6). We enable users to investigate feature distributions by providing large, interactive histograms for each feature for the entire dataset, as seen in Figure 5.3. These histograms treat all features as categorical and when a user hovers over a bar, a tooltip shows the value of this category and how many instances there are with that value in the entire dataset. Furthermore, clicking on one of the rows reveals a collapsible view of all the possible values for the feature. Users are also able to hover over the expanded values to see their location in the histogram.

Subgroup generation. The *Feature Distribution View* also allows users to generate user-specified subgroups. Model developers are often aware of certain intersectional subgroups for which they want to ensure fairness (C1). We define a subgroup as a subset

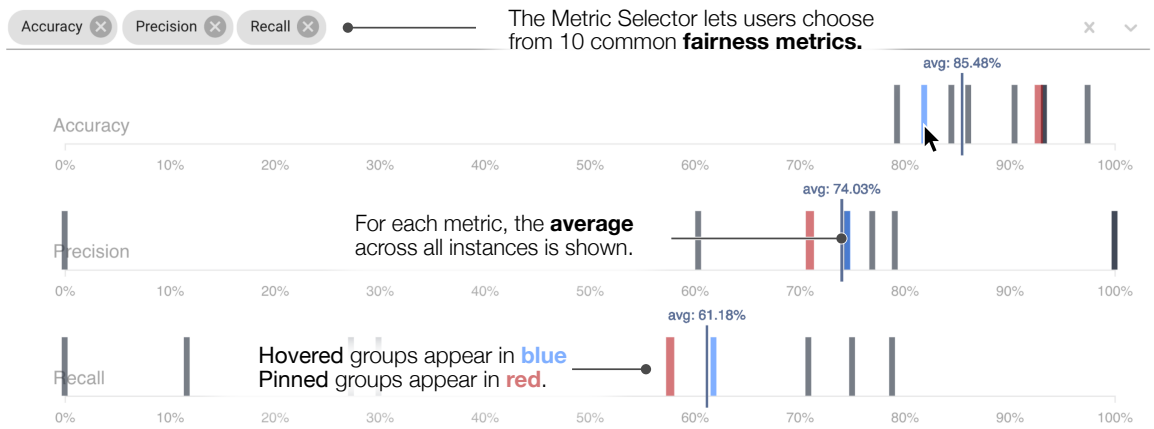


Figure 5.4: In the *Subgroup Overview*, users can see how different subgroups compare to one another according to various performance metrics. As more metrics are selected at the top, additional strip plots are added to the interface. Here, a user has **pinned** the Female subgroup and **hovers** over the Male subgroup.

of a dataset in which all instances share certain values, e.g., the subgroup of blue circles in Figure 5.2.

Our interface allows users to generate both specific subgroups and all subgroups of multiple features by selecting a combination of features and values. For instance in Figure 5.3, if a user checks the feature "race" and "sex", then mutually exclusive subgroups will be generated out of all the instances in the dataset divided on their values for "race" and "sex". However, if a user wants to investigate a particular subgroup, they can select a specific value for "race" and "sex" to add a subgroup of all instances with those specific values. Users can pick any number and combination of features and values by which to define their subgroups, and thus are at liberty to define how general or specific the subgroups they want to explore are.

5.4.2 Subgroup Overview

Once a user has generated subgroups, they should be able to understand which subgroups the model is underperforming on across various metrics and further investigate interesting

subgroups (C2). The *Subgroup Overview* provides a high-level view of this information as multiple interactive and dynamic strip plots (C2).

When a user clicks the "Generate Subgroups" button (Figure 5.3), *FairVis* splits the data into the specified subgroups and calculates various performance metrics for them. These groups are then represented in the multiple strip plots as lines corresponding to their performance for the respective metric.

Visualizing multiple fairness metrics. Due to the inherent tradeoffs between different fairness requirements as shown by the *impossibility theorem*, users must choose which metrics they want to prioritize and investigate (C5). To facilitate this interaction, we allow users to select which metrics are displayed in the *Subgroup Overview* by adding and removing performance metrics through the bar seen at the top of Figure 5.4. Selecting a new metric adds an additional strip plot for that metric with all the current subgroups. We also show the corresponding dataset average per metric in each strip plot to provide context as to how each subgroup is doing in relation to the overall dataset.

In total, users can select from the following metrics: Accuracy, Recall, Specificity, Precision, Negative Predictive Value, False Negative Rate, False Positive Rate, False Discovery Rate, False Omission Rate, and F1 score. These metrics were selected as they are typically the most common metrics used for evaluating the equity and performance of classification models. The performance metrics are derived from the same base outcome rates of true positives, true negatives, false positives, and false negatives. If users find that they need different metrics for performance, they can add a new definition using the base rates which are available in the system.

When a user hovers over a subgroup in a strip plot, the corresponding group is highlighted on every plot currently displayed. This allows users to see how an individual group performs on several different metrics at once [C2, C5]. To further investigate a subgroup, the user can click on a bar to pin the group and use the *Detailed Comparison View* to further investigate the group.

Choice of visual encoding. We chose a strip plot to visualize performance metrics since it allows users to focus on the relative magnitude of subgroup performance in relation to other subgroups and the overall dataset performance. By juxtaposing plots, users are able to see how different metrics are spread out [63]. One of the shortcomings of strip plots is that they can become crowded and hard to use with a large number of subgroups. We address this issue by allowing users to filter the strip plot by subgroup size. While subgroups come in all sizes, groups that are only a few instances are usually not statistically significant enough to draw conclusions from. The size filtering mechanism can help users narrow their search space (C3) and improve the functionality of the strip plot.

While designing our system we considered different visual encodings for displaying subgroups, especially a scatterplot matrix. We decided to use a strip plot over a scatterplot matrix for several reasons. First, since each of the performance metrics is derived from the same base rates, many of the relationships between metrics are arithmetic and not indicative of interesting patterns. We investigated outliers and found that they did not systematically represent any interesting subgroups. Additionally, scatterplot matrices redundantly encode information, as every metric is displayed multiple times. Our strip plot implementation only includes each metric once while still allowing users to see how the group performs in regards to other metrics. Multiple strip plots allow us to display the most important information in a clean and understandable manner; namely, how a given subgroup is performing for selected metrics and in relation to the overall dataset and other subgroups.

5.4.3 Suggested Subgroups

While many users may know of certain groups in their dataset they need to ensure fairness for, it is possible that the model developer has little domain knowledge and does not know where to start. Since there are a combinatorially large number of subgroups in a dataset, it is daunting and often times not feasible to manually inspect groups for every combination of features.

To help the user find potentially biased subgroups, we generate subgroups algorithmically and present them to the user for investigation. The *Suggested and Similar Subgroup View* at the bottom of the interface displays these subgroups and allows the user to sort them by any fairness metric to discover underperforming subgroups (C3).

Generating and Describing Suggested Subgroups

To create the suggested subgroups, we use a clustering-based generation technique. By clustering instances, we can generate groups with significant statistical similarity that can be described by a few dominant features. We can subsequently calculate their performance metrics and display them to the user.

We first cluster all the data instances by their feature values in one-hot encoded form. We use the well-known k-means clustering algorithm as our clustering algorithm [73] with k-means++ as the seeding [16]. Users are able to choose the hyperparameter k to balance the number and size of generated subgroups — a smaller k produces larger, less defined groups while a larger k has the opposite effect. Users run the clustering as a pre-processing script before uploading their data to *FairVis*.

We also experimented with more sophisticated clustering algorithms like the density-based algorithms such as DBSCAN [58] and OPTICS [15], which can generate arbitrarily shaped and sized clusters. While the statistical quality of the density-based clusters can be higher, we found that the flexibility provided by allowing users to modify k is more helpful for discovering important and useful subgroups. Additionally, we found that since we were clustering on many one hot encoded categorical features, DBSCAN's notion of density was not as useful and k-means produced higher quality clusters. Given prior successful application of k-means to a variety of problems and tasks with both categorical and numerical features, we decided to first adapt k-means for *FairVis*.

Once the clusters have been generated, the makeup of the group must be described to the user. A cluster's instances are made up of a variety of values for each feature, but

some features may be more dominated by one value than others. We define a *dominated feature* as a feature that consists of mostly one value, the *dominant value* in a subgroup. For example, if a cluster is 99% male for the feature sex, sex is a dominated feature with a dominant value of male.

The most dominant features can be used to describe the makeup of a subgroup to the users. We rank how dominant features of a group are by calculating the entropy of each feature distribution over its values. Entropy is used since it describes how uniform a feature is. The closer a feature's entropy is to 0, the more concentrated the feature is in one value, making it more dominant in that subgroup.

We formalize the technique for finding dominant features as follows. Suppose we have a set of features, $\mathcal{F} = \{f_1, f_2, ..., f_i, ...\}$, with each feature, f_i, having a set of possible values, $V_i = \{v_{i1}, v_{i2}, ...\}$. We calculate the *feature entropy* for the k-th subgroup and i-th feature, $S_{k,i}$, as follows:

$$S_{k,i} = -\sum_{v \in V_i} \frac{N_{k,v}}{N_k} \log \frac{N_{k,v}}{N_k}, \tag{5.1}$$

where N_k is the number of instances in the k-th subgroup, and $N_{k,v}$ is the number of instances in the k-th subgroup with value v. For example, if all the instances of subgroup k have value $v_{3,1}$ (e.g., India), for the feature f_3 (e.g., native country), the feature entropy is 0 and f_3 is a dominant feature for the subgroup.

Displaying Suggested Subgroups

We display the generated subgroups in the *Suggested and Similar Subgroup View* at the bottom of the interface, as seen in Figure 5.5. Since the generated subgroups are not strictly defined by a few features, it is important to show the feature distributions for each feature in a group. Each suggested subgroup has a list of its features and dominant value, along with a histogram of the value distribution for each feature. The features are sorted according to their dominance, with the dominant value being displayed under the feature name. This

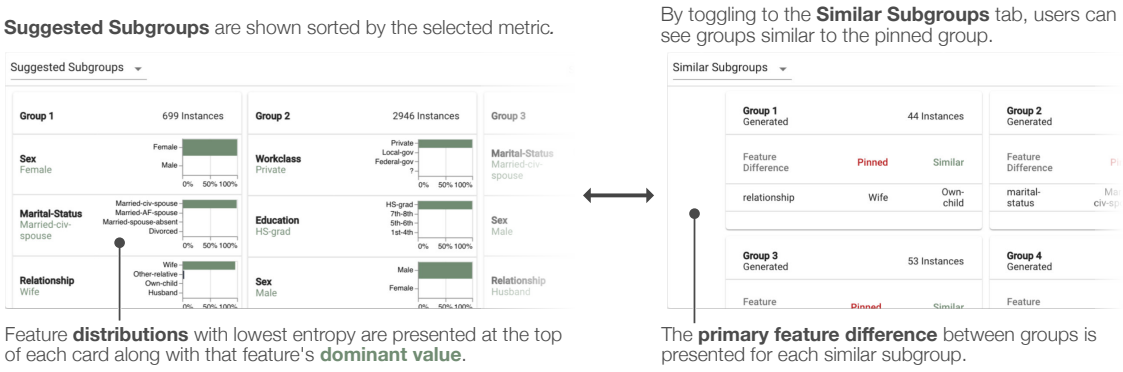


Figure 5.5: Here we can see the *Suggested and Similar Subgroup View* for both suggested and similar subgroups. Users can hover over any card to see detailed feature and performance information in the *Detailed Comparison View*

interface allows users to see what values make up a subgroup and develop an idea of which subgroups may be underperforming.

To explore the groups, users can filter and sort the groups to refine their search space (C3). Since users may find certain metrics more important than others for certain problems, they can choose which metrics to sort the suggested groups by in ascending order (C5). For example, if for a given problem recall is an important metric, users can find generated subgroups with the lowest recall.

Furthermore, users can use the same size slider used to filter the *Subgroup Overview* by size to filter the generated subgroups. Similar to the reasoning for filtering by size in the strip plot, very small groups may not be large enough to draw statistically significant conclusions from. Filtering the groups can remove noise and help users further refine their search space of problematic groups.

Users can hover over a suggested subgroup card to show its detailed performance metrics in the *Detailed Comparison View* and add the group to the *Subgroup Overview* If a user wants to investigate the group further, they can click on the card, pinning the group and allowing them to compare it to other groups or export it for sharing.

5.4.4 Similar Subgroups

Once a user has discovered an interesting subgroup, it can be helpful to look at similar subgroups to either investigate the impact of certain features or to find more general groups with performance issues (C4). Finding similar groups is difficult since it is not a well defined task and can require searching a combinatorially large space.

To formalize similarity and refine the subgroup search space, we apply ideas from statistics and machine learning explainability to this task. When comparing suggested subgroups, we use similarity in the form of statistical divergence to compare how closely related groups are. For user-specified subgroups, we apply the concept of counterfactual explanations by finding groups with minimal value differences that have significantly different performance.

Finding Similar Subgroups

Similarity between subgroups can be thought of as the statistical distance between the feature distributions of groups; the more values two subgroups share, the more similar we consider them. Statistical distance can be measured in a variety of ways, but we found Jensen-Shannon (JS) divergence to be a good measure for our use case. As a derived form of Kullback-Leibler divergence, JS divergence is a similar measure with the benefits of being bi-directional and always having a finite value. Since we often have zero-probability values, JS divergence makes calculating statistical similarity more straightforward and standardized.

We calculate similarity between groups by summing the JS divergence between all features for a pair of subgroups. This sum gives us a measure of how similar two subgroups are on aggregate. Formally, we calculate the total distance D between subgroups k and k'

as follows, where $G_{k,f}$ represents the value distribution of feature f in subgroup k:

$$D(k, k') = \sum_{f \in \mathcal{F}} \text{JS}(G_{k,f} || G_{k',f}). \tag{5.2}$$

This definition of subgroup similarity applies most directly to the suggested subgroups that have some distribution over values for each feature. When comparing two suggested subgroups against each other, we can use the formal definition of JS divergence and sum the average distance of their feature distributions. For comparing user-specified and suggested subgroups against each other we can use a similar technique with a small optimization — since user-specified subgroups will have 0 probability for all values but the selected values in each feature, it is only necessary to calculate the JS divergence for the values present in the user-specified group.

User-specified subgroup comparison. The final potential case for comparison is between two user-specified subgroups. The use of JS divergence as a measure of similarity begins to break down and lose its utility for this use case. The divergence will only ever be one when groups have the same value for a feature or zero when they do not. This metric in practice just counts the number of features with the same value between two groups. While this measure provides some information about subgroup similarity, it is not as informative or accurate as it is when comparing distributions over features in the other two cases.

To provide a more useful comparison of groups, we use the idea of counterfactual explanations [176] which are usually presented in the following form: What are the minimum number of features we have to change to switch the classification of an instance?

Since we are looking at subgroups of multiple instances instead of individual examples, we use a modified notion of counterfactuals for comparing user-specified subgroups: If we only switch one or two feature values for a subgroup, which similar groups have the most surprising changes in performance? This question can help users answer similar questions as they would for the groups found using JS divergence.

Displaying Similar Subgroups

Once similar subgroups have been found for a selected subgroup, we reuse the *Suggested and Similar Subgroup View* from Section 5.4.3 to display the groups to the user. Each subgroup is represented by a card containing a group number and the size of the subgroup. Since selecting a subgroup displays its information in the *Detailed Comparison View* only the information most pertinent to deciding which subgroup to investigate should be displayed.

Continuing with the philosophy of treating similar groups as counterfactuals, we display the primary feature difference between two groups in the case of user-specified subgroups, and the most divergent feature for suggested subgroups. By displaying the feature difference, we emphasize the importance of that feature in the performance difference between the groups.

The same two primary interactions are available for exploring similar groups: sorting and filtering (C3). Users can sort the groups by any fairness metric and filter the groups by size. As with the strip plot and suggested views, this mechanism helps users find statistically significant subgroups that the model is underperforming for in metrics the user finds important.

Similar subgroup importance. Similar subgroups can be informative in two primary manners: finding features which are important for performance and discovering more general subgroups. Given that we are looking at two similar subgroups, they likely only differ in one or two features. If the performance between these two groups is vastly different, it is indicative that the features which are different may contribute significantly to performance (C6). On the other hand, if the two groups have very similar performance, it may mean that a broader subgroup not split using the differing features is also underperforming and should be analyzed.

5.4.5 Detailed Subgroup Analysis and Comparison

The final step in discovering and formalizing group inequity is to examine the details of a subgroup's features and performance. We enable this interaction with the *Detailed Comparison View* on the right hand side of the system.

A user is able to see the details for two groups in the *Detailed Comparison View* the pinned and hovered group. A group can be pinned when a user clicks on it in the *Subgroup Overview* or *Suggested and Similar Subgroup View* and is designated by a light red across the UI. The hovered group is designated by a light blue across the UI. These two distinct colors allow users to see a selected group's information across various different views.

There are three primary components in the *Detailed Comparison View* as seen in Figure 5.6. The topmost component is a bar chart displaying how a group performs for selected performance metrics. While users can see the values of the fairness metrics in the strip plot, the bar chart allows users to see the specific values and enables comparison between groups with a grouped bar chart (C5). The grouped bar chart also enables direct comparison between the pinned and hovered subgroups without the distraction of other groups.

The second component in the *Detailed Comparison View* is a bar chart for the ground truth label balance of both selected subgroups. The label imbalance is important because it can often explain extreme values for metrics like recall and precision and can suggest reasons for bias (C6). For example, a subgroup with 95% negative values can get a 95% accuracy by classifying everything as negative, even though it will have a 0% sensitivity.

The final subgroup comparison interface is a table delineating and comparing the features of the pinned and hovered subgroups. For user-specified subgroups, this table shows the features and values that define the subgroup. For suggested subgroups, this shows the top 5 dominant feature values for that group, and users can see the full distribution in the *Suggested and Similar Subgroup View* view.

Subgroup feature distributions. There is additional information about the pinned and

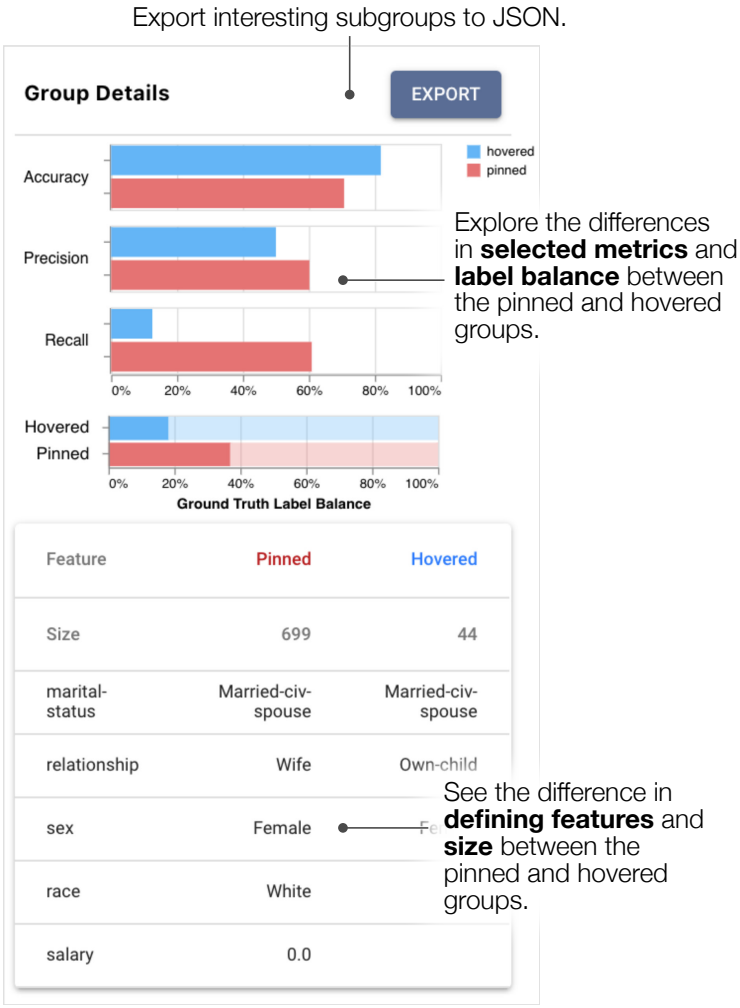


Figure 5.6: In the *Detailed Comparison View*, users can compare the performance and makeup of the pinned and hovered subgroups, providing insight into the causes of performance differences.

hovered subgroup in the *Feature Distribution View* When a subgroup is hovered or pinned, a histogram of each feature's distribution for that group is overlaid on the overall distribution (C2). When there is both a pinned and hovered subgroup, the histograms are overlaid with opacity, allowing users to see how similar the distributions are (Figure 5.7).

The distribution of a subgroup's features can be an important indicator of why a subgroup is underperforming and suggest potential resolutions (C6). If a subgroup's ground truth labels are well balanced, there should be some diversity in the other features of a subgroup for the classifier to be able to discriminate between the two labels. For example,

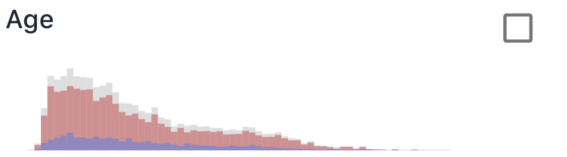


Figure 5.7: When groups are **pinned** and **hovered**, users can compare their feature distributions in the *Feature Distribution View* .

if all White males are also high school educated, married, and from the United States, and they are split between positive and negative classes, it is nearly impossible for a classifier to accurately predict the class for anyone in that subgroup.

An extra interaction in the *Detailed Comparison View* is an export button for sharing a discovered subgroup. Once a user has found subgroups of interest, they can export the pinned and hovered subgroups to a JSON file with their composition and metrics.

5.5 Use Cases

In this section, we describe how *FairVis* can be used in practice to audit models after they have been trained with two example usage scenarios. The first scenario highlights how *FairVis* can be used to audit models for biases against known vulnerable groups in the context of a recidivism prediction system. The second use case shows how users without previous knowledge or intuitions about potential biases can use the system to find issues, for this example with an income prediction model. Both of these use cases utilize real world datasets to demonstrate the applications of our system.

5.5.1 Auditing for Known Biases in Recidivism Prediction

For our first example use case, we will demonstrate how *FairVis* could be used to discover biases in a classifier for recidivism prediction used in the context of deciding who should be given bail. In this use case, we use a classifier based on data gathered by ProPublica

about the real-world tool, COMPAS, that assigns risk scores to criminals to determine their likelihood of re-offending.[1] The original dataset ranks risk from 1-10, with risks from 1-4 constituting ”low” risk, those from 5-7 constituting ”medium” risk, and those from 8-10 as ”high” risk. Following the same methodology as in the ProPublica analysis, we formulate this as a binary classification task by taking risk scores above ”low” (i.e. above 4) as positive model predictions to re-offend, and those at 4 or below as negative predictions as any prediction of risk above low indicates COMPAS is predicting recidivism. Ground-truth labels correspond to whether a defendant released on bail was arrested for another crime within 2 years of their release. An audit by ProPublica revealed that the COMPAS tool is biased to give higher risk scores and thus predict a higher rate of recidivism for Black defendants than other races [14]. Here, we will demonstrate how a data scientist auditing their model in *FairVis* could arrive at the same conclusion.

Known subgroup auditing. To begin their audit, a data scientist would load the COMPAS dataset along with model predictions and ground truth labels into *FairVis*. Given their domain knowledge, the data scientist is aware that, in previous applications involving recidivism prediction, many tools have displayed imbalanced performance for certain genders and races.

To test whether differing performance holds for this model and dataset, the data scientist uses the *Feature Distribution View* to generate all intersectional subgroups of race and sex. When the groups are added to the *Subgroup Overview* (Figure 5.1B), she immediately sees that the groups are spread out broadly across various metrics, suggesting this model may have very different predictive performance on different subgroups. For instance, as we can see in Figure 5.1B (top row), the different intersectional subgroups of sex and race have accuracies ranging from around 50% to 100%.

While the data scientist is interested in the accuracy of her model, she cares most about

[1]COMPAS Recidivism Risk Score Data and Analysis, `https://www.propublica.org/datastore/dataset/compas-recidivism-risk-score-data-and-analysis`

whether her model has large intra-group variation in terms of its false positive rate. For this model, this translates to how many of the people who are not risky are classified as risky. Additionally, she wants to know if these mistakes are distributed unevenly across the different demographic groups. A high false positive rate for this model indicates that many low-risk people (who might be good candidates for release on bail) would be labeled as high-risk by the model. If this model were used to help determine whether a person was seriously considered for release, false positives would correspond to low-risk candidates for release who might be passed over for bail.

To audit the false positive performance metric, the data scientist adds a strip plot for it using the metric selector shown at the top of Figure 5.1B. She then hovers over the bar in the false positive rate strip plot (the bottom row in Figure 5.1B) with the highest value, and sees that this corresponds to the African-American males subgroup with a 43% false positive rate (colored in blue) compared to the dataset average of around 29%. The data scientist pins this subgroup by clicking on this group's strip in the *Subgroup Overview* to investigate it further and compare it to other groups.

By hovering over the other subgroups, she can compare the base rate of recidivism for the pinned group of African-American males relative to other groups. Looking at the Ground Truth Label Balance in Figure 5.1C, we see that the base rate for African-American Males (blue) is almost 60% positive (i.e. 60% rate of recidivism in ground truth), whereas for Caucasian males (red) it is just over 40%.

Thus, if a model makes only one prediction for the entire subgroup of African-American Males, choosing to label the subgroup as positive (a prediction of high recidivism risk) will have higher accuracy than for other subgroups. Less extreme versions of this statement may still hold: to maximize accuracy for this subgroup, a model will use a larger number of positive labels than negative labels. Since the data scientist has noticed that the African-American male subgroup has a very high base rate, but also the highest False Positive Rate out of any of the subgroups in view and still has an accuracy very similar to that

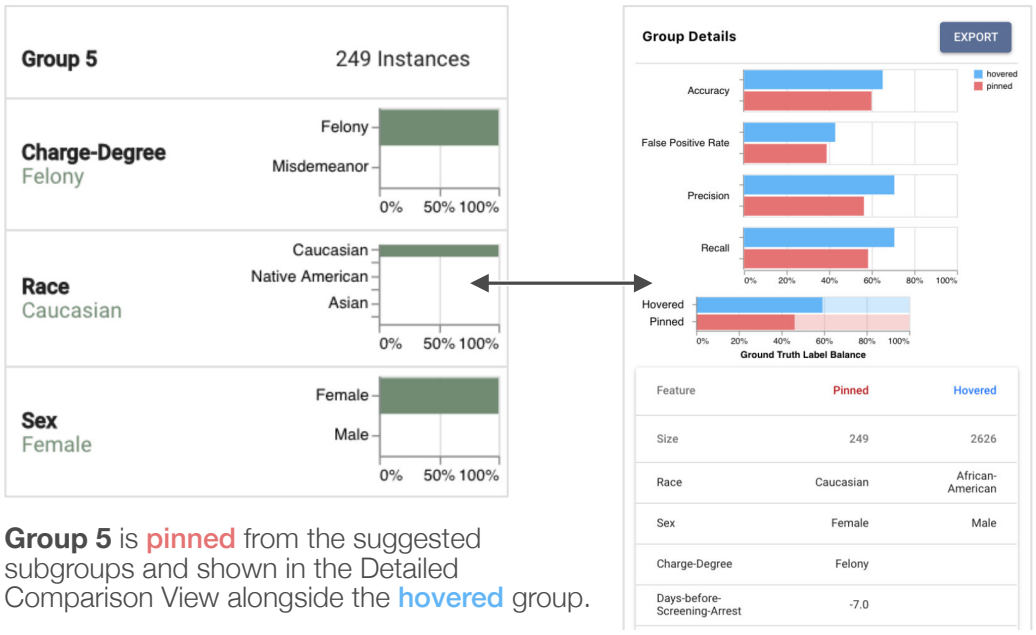


Figure 5.8: A user investigates an interesting subgroup discovered in the *Suggested and Similar Subgroup View*

of Caucasian Males, she thinks this part of her model needs to be altered to give more equitable results.

Here, our example data scientist had suspicions about groups the model might be biased against and was able to leverage *FairVis* to empirically confirm these suspicions. From here, she could use the export function in the system to save these subgroups and devise a plan for corrective action for this model or dataset.

Investigating Suggested Subgroups. Although our data scientist was able to use her domain knowledge to inform her subgroup selection at first, she is interested in whether the model also contains biases against other intersectional subgroups. To aid in the exploration, this data scientist would turn to the *Suggested and Similar Subgroup View* panel to find other potentially problematic groups.

The data scientist first sorts the suggested groups by their false positive rate, since she is most worried about that metric. While the first few groups with the highest false positive rate are made up of African-American males, corroborating her earlier findings, one of the

following groups provides a different result.

The fifth generated group (Figure 5.8) is relatively large with 249 instances, and has a high false positive rate of 39%. By inspecting the composition of this group in the *Detailed Comparison View*nd the subgroup card, she sees that the most defining characteristics of this group are Caucasian females with a felony charge. The label imbalance for this group is about 45% positive and 55% negative and therefore not as pronounced as the base rate imbalance for African-American males (Figure 5.1C). This gives the data scientist two potential hypotheses about sources of this high false positive rate. Her first hypothesis is that the rather small group was not large enough to have been given priority in training; the second is that the class of models considered during training may have been too simple to express the difference between classes in this subgroup. These observations allow our data scientist to make more informed decisions in how to best change her model to address these disparities.

5.5.2 Discovering Biases in Income Prediction

Next, let us consider a model used to offer loan forgiveness to individuals based off their annual income. Our data scientist in this situation does not have access to people's annual income so hopes to use demographic information to predict income. She therefore trains a model on the UCI Adult Dataset[2] to predict whether or not someone makes under $50,000 a year, allowing her to allocate loan forgiveness to lower income candidates with higher fidelity.

Model training. After testing different types of models and hyperparameters, our data scientist finds that a two-layer neural network performs best, with an overall accuracy of 85%. While encouraged by the high accuracy of her model, the data scientist is aware of recent news of algorithmic bias and wants to ensure that her model is treating different

[2] `https://archive.ics.uci.edu/ml/datasets/adult`

demographic groups with similar predictive performance. She decides to audit her model using *FairVis*, and loads her dataset, labels, and model predictions into the system.

Dataset exploration and subgroup creation. When first opening *FairVis*, the data scientist uses the *Feature Distribution View* on the left to look at how balanced her dataset is. While she is unaware of any biases in her data, she immediately notices from looking at the feature histograms that the dataset has a disproportionate representation of males, with males making up more than two-thirds of all instances (see Figure 5.3). To investigate the impact of this imbalance, she selects the feature for sex to generate male and female subgroups. When looking at these two subgroups, she sees in the *Subgroup Overview* that there is a gap of almost 10% in model accuracies between the male and female subgroups (top of Figure 5.4). Despite the higher accuracy of the female subgroup, she notices that the male subgroup has a higher value for precision and recall.

Suggested subgroups. After seeing the fairly large gap in the accuracy of her model between subgroups defined by just one feature, the data scientist is curious about what other combinations of features might lead to poor performance in her model. She turns to the *Suggested and Similar Subgroup View* to see what she can find. Keeping the default sorting of groups by lowest accuracy, she notices that suggested Group 1 (shown on the left side of Figure 5.5) has an accuracy of around 71%, far below the dataset average of 85%. By inspecting the feature distribution charts in the *Suggested and Similar Subgroup View* she sees that this group is primarily defined by Females with a marital status of "Married-civ-spouse" and relationship status of "Wife" as shown by the value distribution graphs in Group 1 of Figure 5.5. Since she wants to better understand why her model is performing poorly for this group, the data scientist tries exploring similar groups.

Similar subgroups. Using her discovery from the Suggested Subgroups tab, our data scientist wants to see how groups of females compare to one another across the "marital-status" and "relationship" features. She generates these subgroups in the *Feature Distribution View* and pins suggested subgroup 1 from earlier to inspect similar groups. Here,

she notices that the similar group with the lowest accuracy is the one comprised of females with a marital status of "married-civ-spouse" but a relationship of "own-child". This group is quite small with only 44 instances.

To see how this group fits into the overall dataset, the data scientist looks to the *Feature Distribution View* Here, she sees that "married-civ-spouse" is the most common value for the Marital-Status feature, and "own-child" is the third most common value for the Relationship feature. These features combine to make a subgroup with relatively few values in the dataset.

When looking at the *Detailed Comparison View* for this similar subgroup, the data scientist notices that the base rate for the "Female, own-child, married-civ-spouse" subgroup is heavily skewed to less than 20% positive ground truth instances (Figure 5.6). The data scientist therefore hypothesizes that the low accuracy for this group may be due to its small size and the skewed base rate. The data scientist notes these observations and aims to gather more data and try using a more expressive model to see if she can address these discrepancies.

5.6 Limitations and Future Work

Improving and measuring the effectiveness of the subgroup generation technique. While we found that the generated subgroups often provide useful suggestions, we hope to test whether these generated groups align well with groups users find important in future work. Collecting labeled data of datasets with outputs and important underperforming subgroups would allow us to quantify the effectiveness of our technique. Additionally, we plan to experiment with more clustering techniques, such as subspace clustering methods [140] to future versions of *FairVis* so that users can see how the groups compare. Especially in high dimensional data, subspace clustering has the potential to reveal interesting groups with poor performance that are primarily defined by only a few features.

Supporting more types of problems and data. *FairVis* currently only supports binary classification and tabular data. The current interface can be expanded to support multiclass classification, but additional visualizations views would need to be added for regression. It would additionally be nice to support some sort of graphical or textual data. The current interface works if the outputs of image classification are loaded with demographic data, but enabling the display of images could aid in auditing groups.

Scaling to millions of instances. The current implementation of *FairVis* is able to scale to tens and hundreds of thousands of data points, but does not support even larger datasets very well. We are looking at improving the efficiency of the subgroup generation and suggestion technique to enable our system to continue to work in browser while at scale.

Suggesting and providing automatic resolutions. Various techniques exist to address bias in machine learning, many of which can be applied as a post-processing step to the output of a classifier. In addition, there are patterns as to what the potential reasons for bias are which could be learned by a model or codified into heuristics. We aim to implement some of the post-processing steps into *FairVis* and add capability to highlight and suggest potential issues.

PART III

LEARNING COMPLEX MODELS BY EXPERIMENTATION

Overview

Recent success in deep learning has generated immense interest among practitioners and students, inspiring many to learn about this new technology. While visual and interactive approaches have been successfully developed to help people more easily learn deep learning, the complexity of modern deep learning models introduces many non-trivial challenges in designing visualization tools for them. The last part of my thesis is on designing and developing interactive educational tools for complex deep learning models with the goal of broadening people's access to learning such models and making sense of complex structure of input datasets. In particular, this part describes two work in this line of research:

- *GAN Lab* (Chapter 6) on understanding deep generative models through interactive experimentation;
- *ETable* (Chapter 7) on browsing and querying complex datasets stored in relational databases.

CHAPTER 6

GAN LAB: LEARNING DEEP GENERATIVE MODELS BY INTERACTIVE EXPERIMENTATION

While many visual and interactive approaches have been successfully helping people, including practitioners, students, and novices, more easily learn deep learning, most existing tools focus on simpler models. This chapter presents *GAN Lab*, the first interactive visualization tool designed for non-experts to learn and experiment with *Generative Adversarial Networks (GANs)*, a popular class of complex deep learning models. With *GAN Lab*, users can interactively train generative models and visualize the dynamic training process's intermediate results. *GAN Lab* introduces new interactive experimentation features for learning complex deep learning models, such as *step-by-step* training at multiple levels of abstraction for understanding intricate training dynamics. Implemented using *TensorFlow.js*, *GAN Lab* is accessible to anyone via modern web browsers, without the need for installation or specialized hardware, overcoming a major practical challenge in deploying interactive tools for deep learning.

6.1 Introduction

Recent success in deep learning has generated a huge amount of interest from practitioners and students, inspiring many to learn about this technology. Visual, interactive methods

This chapter is adapted from work appeared at IEEE VAST 2018 [90].

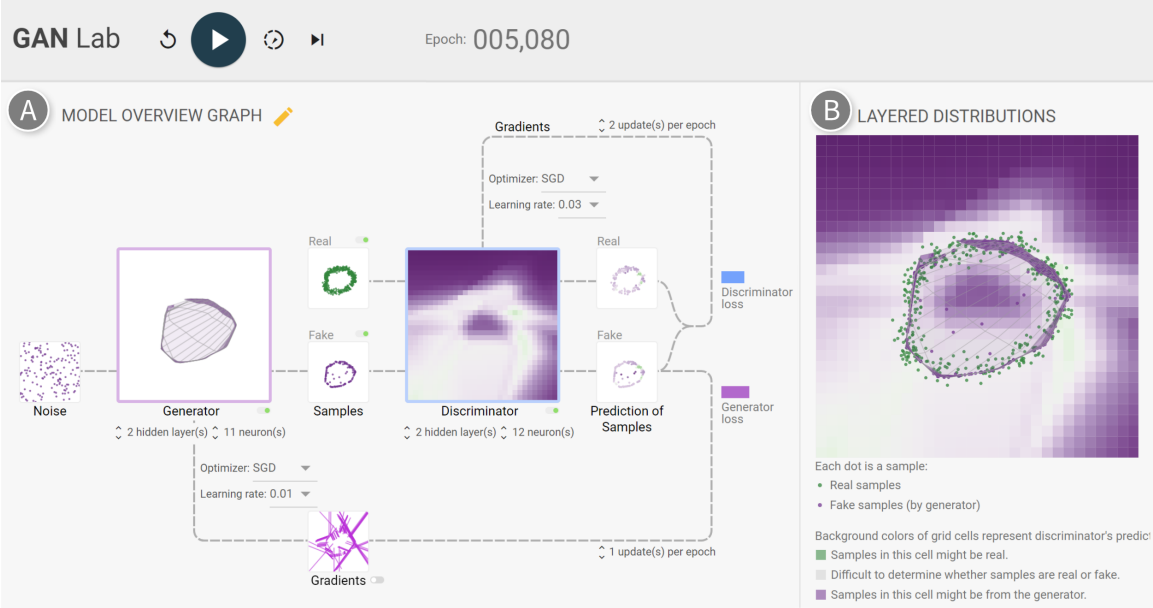


Figure 6.1: With *GAN Lab*, users can interactively train Generative Adversarial Networks (GANs), and visually examine the model training process. In this example, a user has successfully used *GAN Lab* to train a GAN that generates 2D data points whose challenging distribution resembles a ring. **A.** The *model overview graph* summarizes a GAN model's structure as a graph, with nodes representing the generator and discriminator submodels, and the data that flow through the graph (e.g., fake samples produced by the generator). **B.** The *layered distributions* view helps users interpret the interplay between submodels through user-selected layers, such as the discriminator's classification heatmap, real samples, and fake samples produced by the generator.

and tools have successfully been used to describe concepts and underlying mechanisms in deep learning [137, 93, 162, 189]. For example, Karpathy's popular interactive demo [93] enables users to run convolutional neural nets and visualize neuron activations, inspiring researchers to develop more interactive tools for deep learning. Another notable example is Google's *TensorFlow Playground* [162], an interactive tool that visually represents a neural network model and allows users to interactively experiment with the model through direct manipulation; Google now uses it to educate their employees about deep learning [150].

The rise of GANs and their compelling uses. Most existing interactive tools, however, have been designed for simpler models. Meanwhile, modern deep learning models are becoming more complex. For example, *Generated Adversarial Networks (GANs)* [67],

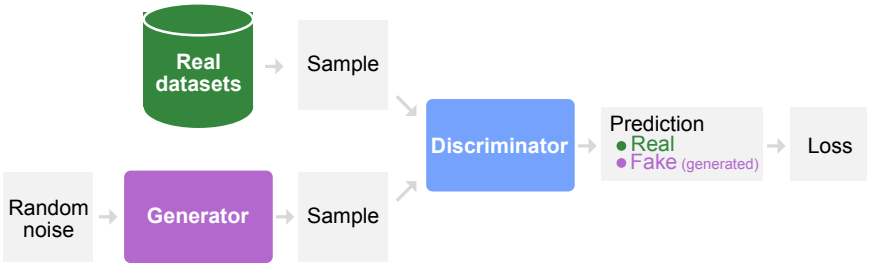


Figure 6.2: A graphical schematic representation of a GAN's architecture commonly used.

a class of deep learning models known for their remarkable ability to generate synthetic images that look like natural images, are difficult to train and for people to understand, even for experts. Since the first GAN publication by Goodfellow et al. [67] in 2014, GANs have become one of the most popular machine learning research topics [75, 108]. GANs have achieved state-of-the-art performance in a variety of previously difficult tasks, such as synthesizing super-resolution images based on low-resolution copies, and performing image-to-image translation (e.g., converting sketches to realistic images) [66].

Key challenges in designing learning tools for GANs. At the high level, a GAN internally combines two neural networks, called *generator* and *discriminator*, to play a game where the generator creates "fake" data and the discriminator guesses whether that data is real or fake (both types of data are mixed together). A perfect GAN is one that generates fake data that is virtually indistinguishable from real data. A user who wishes to learn about GANs needs to develop a mental model of not only what the two submodels do, but also how they affect each other in its training process. The crux in learning about GANs, therefore, originates from the iterative, dynamic, intricate interplay between these two submodels. Such complex interaction is challenging for novices to recognize, and sometimes even for experts to fully understand [152]. Typical architecture diagrams for GANs (e.g., Figure 6.2, commonly shown in learning materials) do not effectively help people develop the crucial mental models needed for understanding GANs.

Contributions. In this work, we contribute:

- ***GAN Lab*, the first interactive tool designed for non-experts** to learn and experiment with GAN models, a popular class of complex deep learning models, that overcomes multiple unique challenges for developing interactive tools for GANs (Section 6.3).

- **Novel interactive visualization design** of *GAN Lab* (Figure 6.1), which tightly integrates a *model overview graph* that summarizes GAN's structure (Figure 6.1A) as a graph, selectively visualizing components that are crucial to the training process; and a *layered distributions* view (Figure 6.1B) that helps users interpret the interplay between submodels through user-selected layers (Section 6.5). *GAN Lab*'s visualization techniques work in tandem to help crystalize complex concepts in GANs. For example, *GAN Lab* visualizes the generator's data transformation, which turns input noise into fake samples, as a manifold (Figure 6.1, big box with purple border). When the user hovers over it, *GAN Lab* animates the input-to-output transformation (Figure 6.3) to visualize how the input 2D space is folded and twisted by the generator to create the desired ring-like data distribution, helping users more easily understand the complex behavior of the generator.

- **New interactive experimentation features** for learning complex deep learning models, such as *step-by-step* training at multiple levels of abstraction for understanding intricate training dynamics (Section 6.6). The user can also interact with the training process by directly manipulating GAN's hyperparameters.

- **A browser-based, open-sourced implementation** that helps *broaden public's education access to modern deep learning technologies* (Section 6.6.3). Training deep learning models conventionally requires significant computing resources. For example, deep learning frameworks, like TensorFlow [2], typically run on dedicated servers. They are not designed to support low-latency computation needed for real-time interactive tools, or large number of concurrent user sessions through the web.

We overcome such practical challenges in deploying interactive visualization for deep learning by using *TensorFlow.js*,[1] an in-browser GPU-accelerated deep learning library recently developed by Google; one of the authors of the paper for *GAN Lab* is a lead developer of TensorFlow.js. Anyone can access *GAN Lab* using their web browsers without the need for installation or specialized backend. *GAN Lab* runs locally on the user's web browser, allowing us to easily scale up deployment for our tool to the public, significantly broadening people's access to tools for learning about GANs. The demo of the tool is available in `https://poloclub.github.io/ganlab/`.

- Usage scenarios (Section 6.8), an observational study (Section 6.9), and a log analysis of the deployed tool (Section 6.10) that demonstrate how *GAN Lab* can help beginners learn key concepts and training workflow in GANs, and assist practitioners to interactively attain optimal hyperparameters for reaching challenging equilibrium between submodels).

VIS's central role in AI. We believe in-browser interactive tools developed by our VIS community, like *GAN Lab*, will play critical roles in promoting people's understanding of deep learning, and raising their awareness of this exciting new technology. To the best of our knowledge, our work is the first tool designed for non-experts to learn and experiment with complex GAN models, different from recent work in visualization for deep learning [118, 168, 117, 86, 143, 183] which primarily targets machine learning experts. Our work joins a growing body of research that aims to use interactive visualization to explain complex inner workings of modern machine learning techniques. Distill, a new interactive form of journal, is dedicated to achieving this exact goal [138]. We hope our work will help inspire even more research and development of visualization tools that help people better understanding artificial intelligence technologies.

[1] TensorFlow.js, `https://js.tensorflow.org`

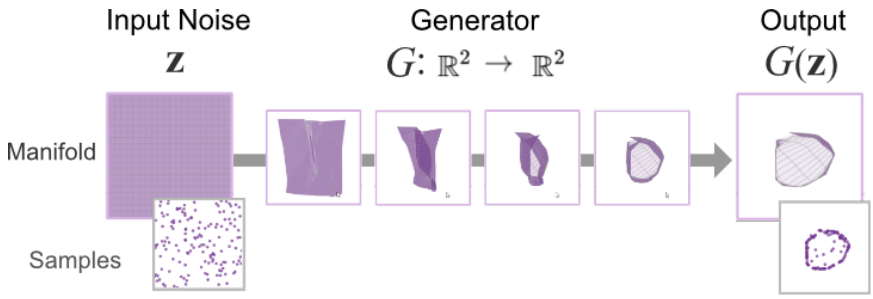


Figure 6.3: In *GAN Lab*, the *generator*'s non-trivial data transformation is visualized as a manifold, which turns *input noise* (leftmost) into fake samples (rightmost). *GAN Lab* animates the input-to-output transformation to help users more easily understand this complex behavior.

6.2 Background: Generative Adversarial Networks

This section presents a brief introduction of Generated Adversarial Networks, which will help ground our discussion in this chapter.

Generative Adversarial Networks (GANs) [67] are a new class of unsupervised generative deep learning models that model data distributions. It can be used for generating multi-dimensional data distributions (e.g., an image is a multi-dimensional data point, where each pixel is a dimension). The model takes *real samples* and random vectors (i.e., *random noise*) as inputs and transforms the random vectors into *fake samples* that mimic the real samples. Ideally, the distribution of the fake samples will be indistinguishable from the real samples. The architecture of GANs is composed of two neural networks, called *generator* and *discriminator*, and is often represented as an abstracted data-flow graph as in Figure 6.2. The generator, G, takes a random noise vector, $\mathbf{z}$, as input and transforms it into a fake sample, $G(\mathbf{z})$ (i.e., a multi-dimensional vector); the discriminator, D, which is a binary classifier, takes either a real or fake sample, and determines whether it is real or fake ($D(\mathbf{x})$ represents the probability that $\mathbf{x}$ is real rather than fake).

A GAN model is iteratively trained through a game between the discriminator and generator. In GAN, two cost functions exist: the one for the discriminator measures the

probability of assigning the correct labels to both real and fake samples (i.e., the sum of $D(\mathbf{x})$ and $1 - D(G(\mathbf{z}))$); the other for the generator measures that for fake samples only (i.e., $1 - D(G(\mathbf{z}))$). The goal of the discriminator is to maximize its cost, but the goal of the generator is to minimize its cost, which introduces conflicts (i.e., zero-sum). Therefore, it has to play a mini-max game to find the optimum. Goodfellow et al. [67] used an interesting analogy to explain how it works, where we can view the generator as a *counterfeiter* who makes fake dollar bills, and the discriminator as the *police*. If the police can spot the fake bills, that means the counterfeiter is not "good enough," so the counterfeiter carefully revises the bills to make them more realistic. As the discriminator (police) differentiates between real and fake samples, the generator (counterfeiter) can glean useful information from the discriminator to revise its generation process so that it will generate more realistic samples in the next iteration. And to continue to receive such helpful information, the generator keeps providing its updated samples to the discriminator. This iterative interplay between the two players leads to generating realistic samples.

6.3 Design Challenges for Complex Deep Learning Models

Our goal is to build an interactive, visual experimentation tool for users to better understand GANs, a complex deep learning model. To design *GAN Lab*, we identified four key design challenges unique to GANs.

C1. [MODEL] **Complex model structures with submodels.** The structures of modern deep learning models (including GANs) are complex; they often incorporate multiple base neural networks or deep learning models as submodels. For example, a GAN combines two neural nets: generator and discriminator; an image captioning model often consists of both CNNs and RNNs for translation between images and text [175]. Effective visualization of such models calls for new strategies different from those designed for conventional models. For example, it is crucial to find the appropriate

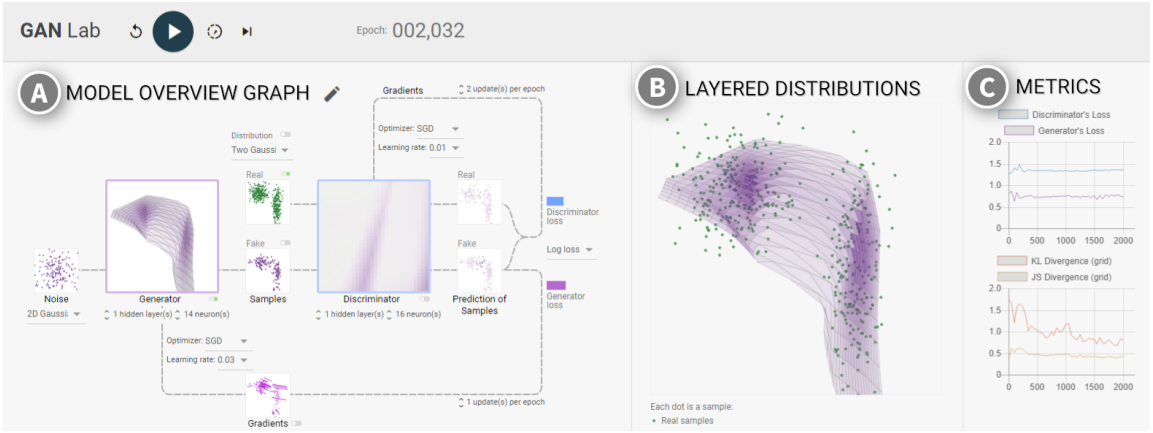


Figure 6.4: The *GAN Lab* interface integrates multiple views: **A.** The *model overview graph* summarizes a GAN model's structure as a graph, with nodes representing the submodels, and the data that flow through the graph; **B.** The *layered distributions* view overlays magnified versions of the graph's component visualizations, to help users more easily compare and understand their relationships; **C.** The *metrics* view presents line charts that track metric values over the training process. Users start the model training by clicking the *play* button on menu bar. The three views are dynamically updated, as training progresses. In this example, real samples are drawn from two Gaussian distributions, and the generator, consisting of a single hidden layer with 14 neurons, has created samples whose distribution is quite similar to that of the real samples.

levels of visual abstraction for the models, as visualizing all low-level details will overwhelm users. Special visual design may be needed to help users interpret the intricate interplay between submodels (e.g., discriminator and generator).

C2. [DATA] **High-dimensional datasets.** As deep learning models often work with large, high-dimensional datasets, visualizing their distributions would quickly create many traditional challenges well-studied in information visualization research [119]. While we may use techniques like *dimensionality reduction* to partially address such issues, this could introduce additional complexities to the systems, potentially distracting users from their main goal of understanding how deep learning models work.

C3. [TRAINING PROCESS] **Many training iterations until convergence.** Deep learning models are trained through many iterations (i.e., at least thousands), introducing non-

trivial challenges for developing interactive tools. As it takes time to converge, the tools need to keep providing users with information during training (e.g., progress), and users may also want to provide feedback to models (e.g., by changing hyperparameters). In addition, while one popular feature used in many experimentation tools is a step-by-step execution of systems [69, 153], the definition of *steps* becomes different in training of complex models, because the training process consists of many iterations and each iteration also consists of the training of multiple submodels.

C4. [DEPLOYMENT] **Conventional deep learning frameworks ill-fitted for multi-user, web-based deployment.** Training deep learning models conventionally requires significant computing resources. Most deep learning frameworks written in Python or C++, like TensorFlow [2], typically run on dedicated servers that utilize powerful hardware with GPU, to speed up the training process. However, even with a powerful backend, they cannot easily support a large number of concurrent user sessions through the web, because each session requires significant computation resources. When combined, even a small number of concurrent sessions can bog down a powerful server. Off-loading computation to the end user is a possible solution, but conventional deep learning frameworks are not designed to support low-latency computation needed for real-time interactive tools.

6.4 Design Goals

Based on the identified design challenges in the previous section, we distill the following main design goals for *GAN Lab*, a novel interactive visualization tool for learning and experimenting with GANs.

G1. **Visual abstraction of models and data flow.** To give an overview of the structure of complex models, we aim to create a visual representation of a model by selectively choosing and grouping low-level operations (and intermediate data) into high-

level components (C1). It helps users visually track how input data are transformed throughout the models. For users to clearly examine the internal model training process and data flow, we would use low-dimensional datasets (C2). (Section 6.5.1)

G2. **Visual analysis of interplay between discriminator and generator.** As GANs internally use two different neural nets, it is important for users to understand how they work together, to get a holistic picture of the overall training process (C1). In response, we would like to enable users to examine and compare the visualizations of the model components to understand they affect each other to accomplish the generation tasks. (Section 6.5.2)

G3. **Dynamic experimentations through direct manipulation of hyperparameters.** We aim to let users dynamically play and experiment with models. To help users quickly understand the roles of many hyperparameters and control them (C3), we would like to design interactive interfaces which users can easily locate and manipulate the options. The users' actions are directly applied to the model training process. (Section 6.6.1)

G4. **Supporting step-by-step execution for learning the training process in detail.** Since the training process of deep learning models consists of many iterations and each iteration also consists of several steps, the step-by-step execution of models can greatly help novices to understand the training process (C3). To address this needs, we aim to design multiple ways to execute models in a step-by-step fashion by decomposing the training process into steps at multiple levels of abstraction. (Section 6.6.2)

G5. **Deployment using cross-platform lightweight web technologies.** To develop a tool that is accessible from multiple users without a need to use specialized powerful backend (C4), we would like to use web browsers both for training models and visualizing results. (Section 6.6.3)

6.5 Visualization Interface of *GAN Lab*

This section describes *GAN Lab*'s interface and visualization design. Figure 6.4 shows *GAN Lab*'s interface, consisting of multiple views. Using the control panel on top, users can run models and control the speed of training, which we describe in detail in the next section (Section 6.6). This section primarily describes the other three views that visualize models and trained results: (A) **model overview graph** view on the left (Section 6.5.1); (B) **layered distributions** view in the middle (Section 6.5.2); (C) **metrics** view on the right (Section 6.5.3). In the figure, 2D real samples are drawn from two Gaussian distributions. The user's goal is to train the model so that it will generate a similar distribution, by transforming 2D Gaussian noise using a neural net with a single hidden layer.

Color scheme. In our visualization, we color real data green and fake data purple. We do not use a more traditional green-red color scheme, as we do not want to associate fake data with a negative value. For visualizing the discriminator, we use blue, a color unrelated to the color scheme chosen for coloring data. For visualizing the generator, we again use the color purple because the generated points are the fake points the model sees.

6.5.1 Model Overview Graph: Visualizing Model Structure and Data Flow

The *model overview graph* view (Figure 6.4 at A) visually represents a GAN model as a graph, by selectively grouping low-level operations into high-level components and presenting data flow among them.

Abstraction of Model Architecture as Overview Graph

The model overview graph visually summarizes the architecture of a GAN model. Instead of presenting all low-level operations and intermediate data (i.e., output tensors), it selectively represents high-level components and important intermediate data as nodes. Specificallly, nodes of the graph include two main submodels (i.e., generator and discriminator)

and several intermediate data (e.g., fake samples). Each submodel, which is a neural network, is represented as a large box, and six data nodes are visualized as small boxes. This decision is based on our observation of how people draw the architecture of GANs [48] (like Figure 6.2). Users are often familiar with the structure of the basic neural networks and more interested in the overall picture and interplay between the two submodels. we place input data nodes on the left side of the submodels and output nodes on the right (for forward data flow). Then we draw edges where forward data paths are drawn from left to right and backward data paths, representing *backpropagation*, are drawn as two large backward loops (one for the discriminator and the other for the generator).

Visualization of Nodes in Overview Graph

We visualize the current states of models within the nodes in the graph for users to understand and monitor the training process.

Using 2D datasets to promote comprehension. One challenge in visualizing this information arises from the difficulty of visualizing a large number of high-dimensional data points. To tackle this issue, we decided that we limit our GAN models to generate two-dimensional data samples, while GANs often work with high-dimensional image data. This decision is mainly for helping users easily interpret visualization and focus to understand the internal mechanisms of the models. As many researchers identified, when designing interactive tools, it is even more desirable to focus on simpler cases [156]. Visualization of two-dimensional space is easier for people to understand how data are transformed by the models than that of higher- or one-dimensional spaces: 3D or larger requires dimensionality reduction techniques that add more complexity to users and hinders their understanding.

Below we describe how we visualize each node. We show a miniaturized copy of each node's visualization from Figure 6.4 for easier referencing.

Real samples are what a GAN would like to model. Each sample, a two-dimensional vector, is represented as a green dot, where its x and y position represents the values of its two-dimensional data point. In this example, two Gaussian distributions exist: on the upper-left, and on the right.

Random noise, an input to the generator, is a set of random samples. In *GAN Lab*, noise can be either 1D or 2D. If it is a 1D value, data points are positioned in a line; if a 2D vector (which is default), positioned in a square box, as shown in the small figure on the right.

Fake samples are output produced the generator by transforming the random noise. Like real samples, fake samples are also drawn as dots, but in purple. For a well-trained GAN, the generated distribution should look indistinguishable from the real samples' distribution.

Generator, a neural net model, is a transformation function, $G : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, that maps a 2D data point (i.e., random noise, $\mathbf{z}$) to another 2D data point (i.e., fake sample, $G(\mathbf{z})$). We visualize the transformed results as a 2D **manifold** [137], as in the figure on the right. To draw this manifold, we first create a square grid (e.g., 20x20) for the random noise (see Figure 6.5, leftmost) where each cell represents a certain noise range (e.g., $\{\mathbf{z} = (z_1, z_2) \mid 0.85 \leq z_1 < 0.90 \wedge 0.10 \leq z_2 < 0.15)\}$). We color each cell in purple, encode its probability density with opacity (i.e., more opaque means more samples in the cell). The generator G transforms the random noise into fake samples by placing them in new locations. To determine the transformation for the grid cells, we feed each cell's four corners into the generator, which returns their transformed positions forming a quadrangle (e.g., $G(0.85, 0.10) = (0.21, 0.75)$, $G(0.85, 0.15) = (0.24, 0.71)$, ...). Thus, the whole grid, now consisting of irregular quadrangles, would look like a warped version of the original regular grid. The density of each (warped) cell has changed. We calculate its new density by dividing the original density value (in the input noise space) by the *area* of the quadrangle. Thus, a higher opacity means

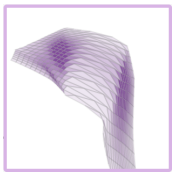


Figure 6.5: Visualization of generator's transformation. When users mouse over the generator node, an animation of the square grid transitioning into a warped version is played.

more samples in smaller space. Ideally, a very fine-grained manifold will look almost the same as the visualization of the fake samples. Our visualization technique aligns with the *continuous scatterplots* idea [17] that generalizes scatterplots to continuous data by computing the density of data samples in the scatterplot space. To help users better understand the transformation, we show an animation of the square grid transitioning into the warped version (see Figure 6.5), when users mouse over the generator node in the overview graph.

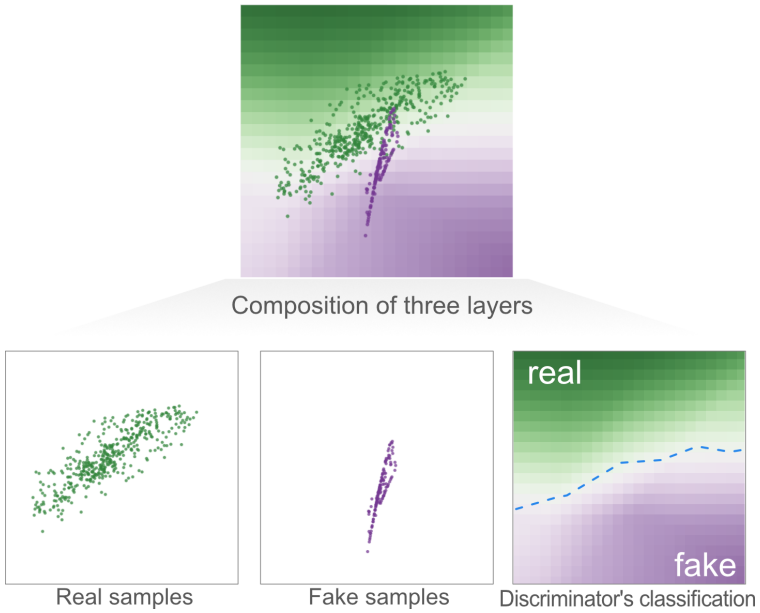


Figure 6.6: The discriminator's performance can be interpreted through the *layered distributions* view, a composite visualization composed of three layers selected by the user: *Real samples*, *Fake samples*, and *Discriminator's classification*. Here, the discriminator is performing well, since most real samples lies on its classification surface's green region (and fake samples on purple region).

Discriminator is another neural net model, which is a binary classifier, that takes a sample as input and determines whether it is real or fake by producing its prediction score (values from 0 to 1). We visualize the discriminator using a 2D **heatmap**, as in TensorFlow Playground [162]. The background colors of a grid cell encode the prediction values (darker green for higher values representing that samples in that region are likely real; darker purple for lower values indicating that samples are likely fake). As a GAN approaches the optimum, the colors become more gray (as in the above figure), indicating the discriminator cannot distinguish fake examples from the real ones.

Predictions are outputs from the discriminator. We place real or fake samples at their original positions, but their fill colors now represent prediction scores determined by the discriminator. Darker green indicates it is likely a real sample; darker purple likely a fake sample. In this example, most samples are predicted as fake, except for the ones on the upper left.

Gradients for generator are computed for each fake sample by backpropagating the generator's loss through the graph. This snapshot of gradients indicates that how each sample should move to, in order to decrease the loss value. As a gradient represents a vector, we visualize it as a line starting from the position of each sample, where length indicates strength.

6.5.2 Layered Distributions: Visual Analysis of Interplay between Discriminator and Generator

In complex models like GANs, it is a key to understanding relationships among several elements of the models. For example, users may want to check how the distribution of fake samples are similar to those of real samples. Although users can perform a side-by-side comparison of the two different nodes on the model overview graph, this task would be

Figure 6.7: Evaluating how well the distribution of fake samples matches that of real samples by turning on real samples' density contour and fake samples in the layered distributions view.

greatly improved when they are overlapped in the same coordinates.

To help visually analyzing relationships among multiple components, we create a *layered distributions* view (Figure 6.4 at B) that presents a large canvas showing the visual representations of the nodes in the model overview graph as multiple layers. The layers can be turned on or off using toggle switches. We do not intend to visualize all layers, as it is overwhelming to users and it is much more effective to include only the useful information for particular tasks. The view currently supports six layers. All layers, except the one for the real samples' density contour, are magnified versions of the visual representations of the graph nodes we described in the previous subsection (Section 6.5.1). The layers are:

- Real samples (green dots)
- Real samples' density contour (see Figure 6.7)
- Generator transformation manifold
- Fake samples (purple dots)
- Discriminator's classification heatmap
- Generator's gradients (pink lines)

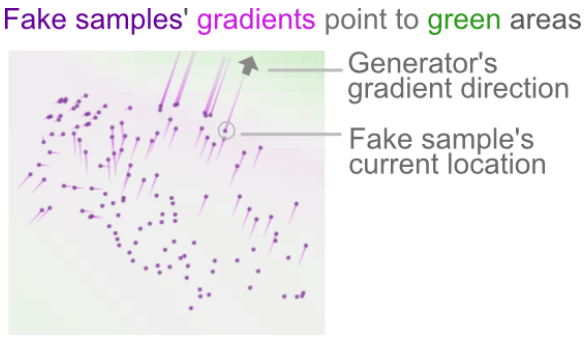


Figure 6.8: Example of understanding the interplay between discriminator and generator using the layered distributions view. Fake samples' movement directions are indicated by the generator's *gradients* (pink lines), based on those samples' current locations and the discriminator's current classification surface (visualized by background colors).

Useful combinations of layers. By selecting which visualizations to be included in the canvas, users can visually analyze the state of the models and the interplay between discriminator and generator, from multiple angles. We describe three example combinations that support multiple analysis tasks. First, Figure 6.6 illustrates that the discriminator may be visually interpreted by comparing the samples' positions with grid's background colors. Here, the discriminator is performing well, as most real and fake samples lie on its classification's green and purple regions, respectively. The second example in Figure 6.7 illustrates how users may visually evaluate how well the distribution of fake samples matches that of the real samples. It helps users to determine whether the two distributions are similar or not, which is the main goal of GANs. The last example in Figure 6.8 shows how the view can help users understand the interplay between discriminator and generator. Fake samples' gradient directions point to the classification's green regions, meaning that the generator leverages information from the discriminator to make fake samples less distinguishable from the real ones.

6.5.3 Metrics: Monitoring Performances

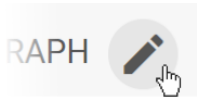
The *metrics view* (Figure 6.4 at C) shows a number of line charts that track several metric values changing as the training promises. *GAN Lab* currently provides two classes of metrics. The first kind is the loss values of the discriminator and generator, which are helpful for evaluating submodels and comparing their strengths. The second kind of metrics is for evaluating how similar the distributions of real and fake samples are. *GAN Lab* provides Kullback-Leibler (KL) and Jensen-Shannon (JS) divergence values [114, 171] by discretizing the 2D continuous space (via the grid). Formally, the KL divergence value is defined as $KL(P_{\text{real}}||P_{\text{fake}}) = -\sum_i P_{\text{real}}(i) \log \frac{P_{\text{fake}}(i)}{P_{\text{real}}(i)}$, where $P_{\text{real}}(i)$ is the probability density of the real samples in the i-th cell, calculated by dividing the number of real samples in the i-th cell by the total number of real samples; $P_{\text{fake}}(i)$ is similarly defined for the fake examples. We decided to use these measures, among others, because they are some of the most commonly used approaches for comparing distributions and they do not incur heavy in-browser computation overhead.

6.6 Interactive Experimentation

This section describes how users can interactively experiment with GAN models using *GAN Lab*.

Basic workflow. Clicking the play button, located on the top of the interface, starts running the training of a GAN model and dynamically updates the visualizations of intermediate results every n epochs (a.k.a., iterations). This helps users keep track of the model's training and examine how they evolve. Users can pause the training by clicking the pause button (the play button changes to pause button during training).

6.6.1 Direct Manipulation of Hyperparameters

GAN Lab is designed for users to directly manipulate model's training as easy as possible. When users click the editing icon on the right side of 

the label for the *model overview graph* view, several up/down buttons or dropdown menus, which controls the model's hyperparameters, are shown (see Figure 6.4). Each item is located near its relevant submodel or data node for users to easily locate it. Users can directly change the values using the buttons or dropdown menus, and the user's actions (e.g., increasing learning rate) are immediately applied to the model training process, except for some of the submodel-specific options (e.g., number of hidden layers), and the effects of this change will be visualized, as the training further progresses. This would greatly help users understand how these hyperparameters affect the model training process. The current available hyperparameters in *GAN Lab* include:

- Number of layers for generator and discriminator
- Number of neurons in each layer for generator and discriminator
- Optimizer type (e.g., Stochastic Gradient Descent, Adam) for updating the generator and discriminator
- Learning rates for updating the generator and discriminator
- Loss function (e.g., log loss [67], least square loss (LS-GAN [125]))
- Number of training runs for discriminator (and generator) for every epoch[2]
- Noise dimension (e.g., 1D, 2D) and distribution type (e.g., uniform, Gaussian)

GAN Lab also allows users to pick a distribution of real samples using the drop-down menu that currently implements five examples (e.g., ring). Users can also specify a new distribution by drawing one on a canvas with brush, as illustrated in Figure 6.9.

[2]In training of GANs, for every epoch, the discriminator and generator are trained by turns. Goodfellow et al. [67] suggested that the discriminator can be updated k more times in practice, and *GAN Lab* enables to adjust this k value.

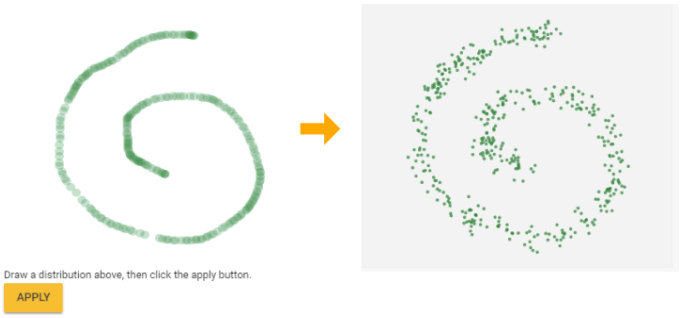
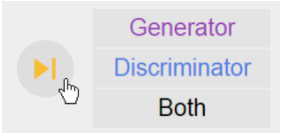


Figure 6.9: Users can create real samples by drawing their distribution.

6.6.2 Step-by-Step Model Training at Multiple Levels

GAN Lab supports *step-by-step* training at multiple levels of abstraction for understanding intricate training dynamics. The step-by-step execution of systems is one of the useful ways for learners to understand how they work [159], however, training of GANs consists of thousands of iterations and each iteration also consists of several steps (as illustrated in Figure 6.10). To address this problem, we decompose the training process into steps in multiple levels: epoch-, submodel-, and component-level.

Manual Step Execution in Epoch-Level



Users can train a model for only one epoch, by clicking a button once. This epoch-level step execution is designed to help users track the training process to see how models update to find the optimum state through iterations. To use this feature, a user first clicks the step icon on top, which will shows three buttons. The last button ("Both") represents the training for one epoch. We describe the other two buttons' usage next.

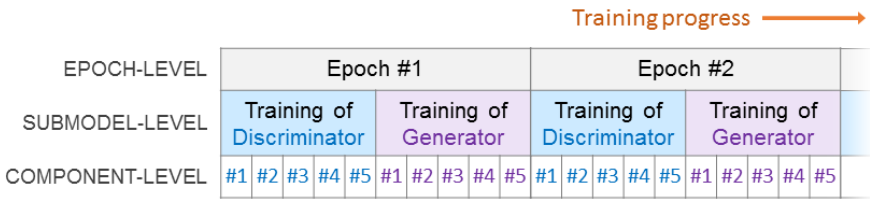


Figure 6.10: Training typically involves of thousands of epochs (iterations). Each epoch includes training both discriminator and generator. *GAN Lab* supports step-by-step model training at different abstraction levels.

Manual Step Execution in Submodel-Level

A single epoch consists of training of a discriminator and z generator, as illustrated in Figure 6.10. *GAN Lab* allows users to update only the discriminator or generator. The experimentation of training only one of the two submodels is effective for users to understand how they work differently. For example, clicking the button for the discriminator changes the background grid while preserving the positions of fake samples. On the other hand, clicking the discriminator button moves the fake samples while fixing the background grid. To use this feature, users click the step icon first, then the three buttons will be shown. The first button is for training the discriminator; the second button is for the generator; and the last button is for training both submodels.

Slow-Motion Mode in Component-Level

GAN Lab also provides the *slow-motion mode*, designed to help novices learn how each component of the model works to make updates within each epoch. It works differently from the manual step execution described in the two previous paragraphs. When users turn on this mode by clicking the icon on top during training, it slows down the speed of training. In addition, two similar lists of five steps are presented: one for updating the discriminator and the other for the generator, as depicted in Figure 6.11. The five steps include (1) running the generator; (2) running the discriminator; (3) computing dis-

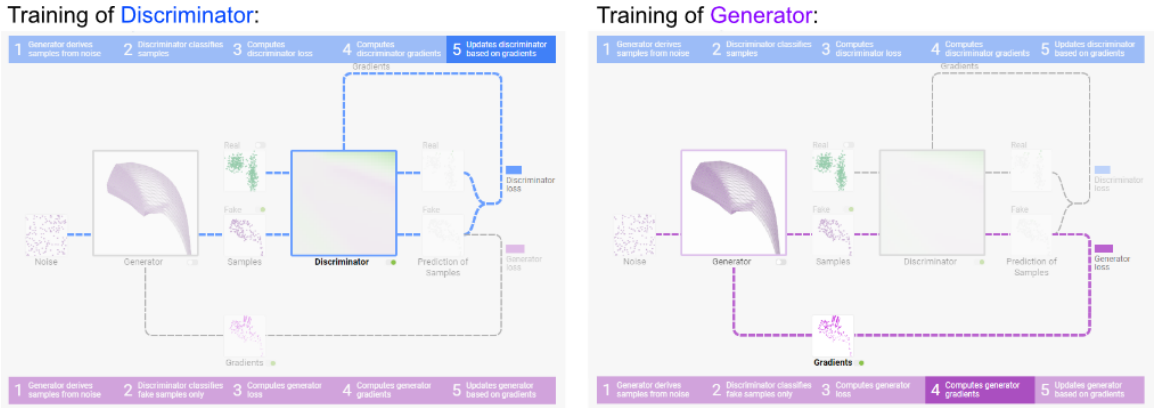


Figure 6.11: The slow-motion mode slowly executes the model training process in a component level, in a step-by-step fashion. The steps are grouped into two lists, one for discriminator and the other for generator, each consisting of five steps.

criminator or generator loss; (4) computing gradients; and (5) updating the discriminator or generator. For every few seconds, it moves to the next step highlighting the corresponding model components with textual descriptions. For example, each of the five steps for the discriminator is highlighted one after another. At the same time, the whole training loop for the discriminator is also highlighted (i.e., edges colored by blue). Once the five steps are completed, it proceeds to the training of the generator, highlighting the training loop for the generator (i.e., purple edges) and executing its five steps. By following these training paths, users can learn how every component is used in training GANs.

6.6.3 Browser-based Implementation for Deployment

GAN Lab is an open-source, web-based visualization tool. Anyone can access it using their modern web browsers without the need for installation or specialized backend. The demo is available at `https://poloclub.github.io/ganlab/`.

The tool is implemented in HTML and TypeScript (a typed version of JavaScript) with a few open-source JavaScript libraries: *TensorFlow.js*[3] is used for training and running mod-

[3]TensorFlow.js, `https://js.tensorflow.org/`

els, which we will elaborate in detail in the next paragraph; *Polymer*[4] is used for building web applications; and *D3.js*[5] is used to visualize the model overview graph and layered distributions. The source code is available in `https://github.com/poloclub/ganlab/`.

Using *TensorFlow.js* for model building and training. *GAN Lab* runs locally on user's web browsers by using *TensorFlow.js* (formerly known as *deeplearn.js*), an in-browser GPU-accelerated deep learning library, developed by Google. The *TensorFlow.js* library uses WebGL to efficiently perform computation on browsers, required for training deep learning models. Not only does it enable rapid experimentation of the models, but also allows us to easily scale up deployment for the public. While most other implementations of GANs that use Python or other server-side languages would backfire when multiple users train models concurrently, our GAN models are trained in JavaScript, which means that that the models and their visualizations run locally on web browsers, enabling us to significantly broaden people's access to *GAN Lab* for learning about GANs.

6.7 Informed Design through Iterations

The current design of *GAN Lab* is the result of 11 months of investigation and development through many iterations. Below we share two key lessons learned from our experience.

The model overview graph is a crucial and effective feature that helps users develop mental models for GANs. Our early design (Figure 6.12) did not include the overview graph. Instead, it displayed a long list of hyperparameters. While that design had all the necessary features for training GANs interactively, pilot users, including machine learning experts, commented that the tool was difficult to use and to interpret. The main reason is that, without an overview, users had to develop mental models for GANs (in their heads)

[4]Polymer, `https://www.polymer-project.org/`
[5]D3.js, `https://d3js.org/`

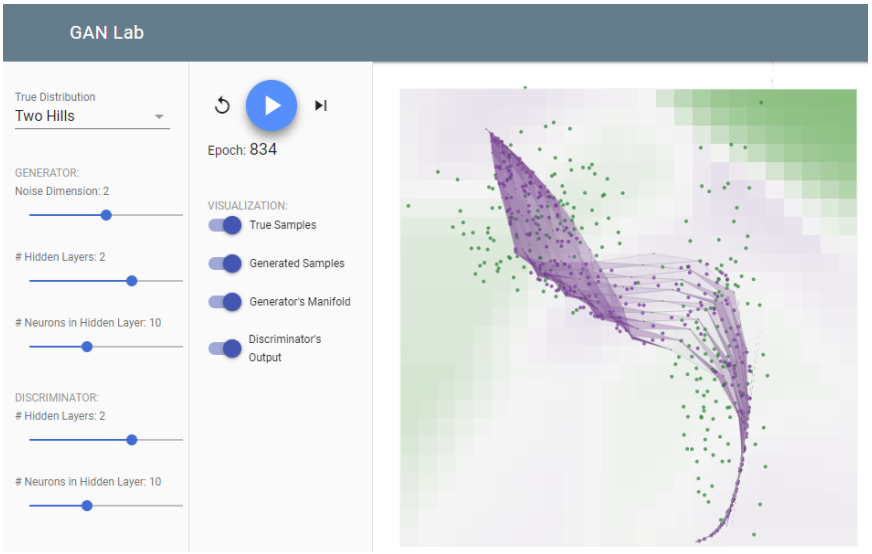


Figure 6.12: Early design of *GAN Lab* did not include a model overview graph that helps users develop mental models for GANs.

to keep track of how the larger number of hyperparameters map to the different model components. This finding prompted us to add the model overview graph, inspired from common architecture diagrams for GANs, which helps users build mental models for the training process of GANs [122].

Animating the generator's transformation (Figure 6.5) was helpful in helping users interpret the manifold visualization. Our early version only showed the transformed manifold (e.g., Figure 6.5, rightmost). However, many users were puzzled by what they saw because, the manifold could be so severely distorted that they could not tell what its original shape was (a uniform 2D grid), thus they could not make the connection to realize that the manifold visualization was indeed representing the generator's output. We though about adding text to the interface to explain the manifold, but as *GAN Lab* is intended to be used as a standalone tool, we would like to keep the visual design compact, and we wanted to include textual descriptions only when necessary. Thus, we came up with the idea of visually explaining the transformation as an animated transition, which was immediately clear to all users.

6.8 Usage Scenarios

This section describes two example usage scenarios for *GAN Lab*, demonstrating how it may promote user learning of GANs. The scenarios highlight: (1) how beginners may learn key concepts for GANs by experimenting with the tool's visualizations and interactive features (Section 6.8.1); (2) how the tool may help practitioners discover advanced inner-workings of GANs, and how it can assist them to interactively attain optimal hyper-parameters for reaching equilibrium between submodels (Section 6.8.2).

6.8.1 Beginners Learning Concepts and Training Procedure

Consider Alice, a data scientist at a technology company, who has basic knowledge about machine learning. Recently, she has started to learn about deep learning, and a few of the introductory articles she has been reading mention GANs. Excited about their potential, she wishes to use *GAN Lab* to interactively learn GANs.

Becoming familiar with basic workflow. When Alice launches *GAN Lab* in her web browser, she sees the *model overview graph*, which looks like a GAN architecture diagram that she has seen in her articles. By default, *real samples* are drawn from a 2D distribution that resembles a line. She clicks the *play* button on the tool bar. During the training, the movement of the fake samples in the *layered distribution* view attracts her attention. They keep moving towards the real samples.

Using the slow-motion mode for tracking the training procedure. Alice is aware that discriminator and generator take turns to train, but she is unsure of what that means. To see how training progresses, Alice clicks the slow-motion icon to enter the *slow-motion* training mode, which slows down the speed of training, and presents two lists of training steps, one for the discriminator, and another for the generator (see Figure 6.11). She notices that in for every epoch, the discriminator is trained first, then the generator follows. The two models' training sequences seem very similar, but she discovers several key differences.

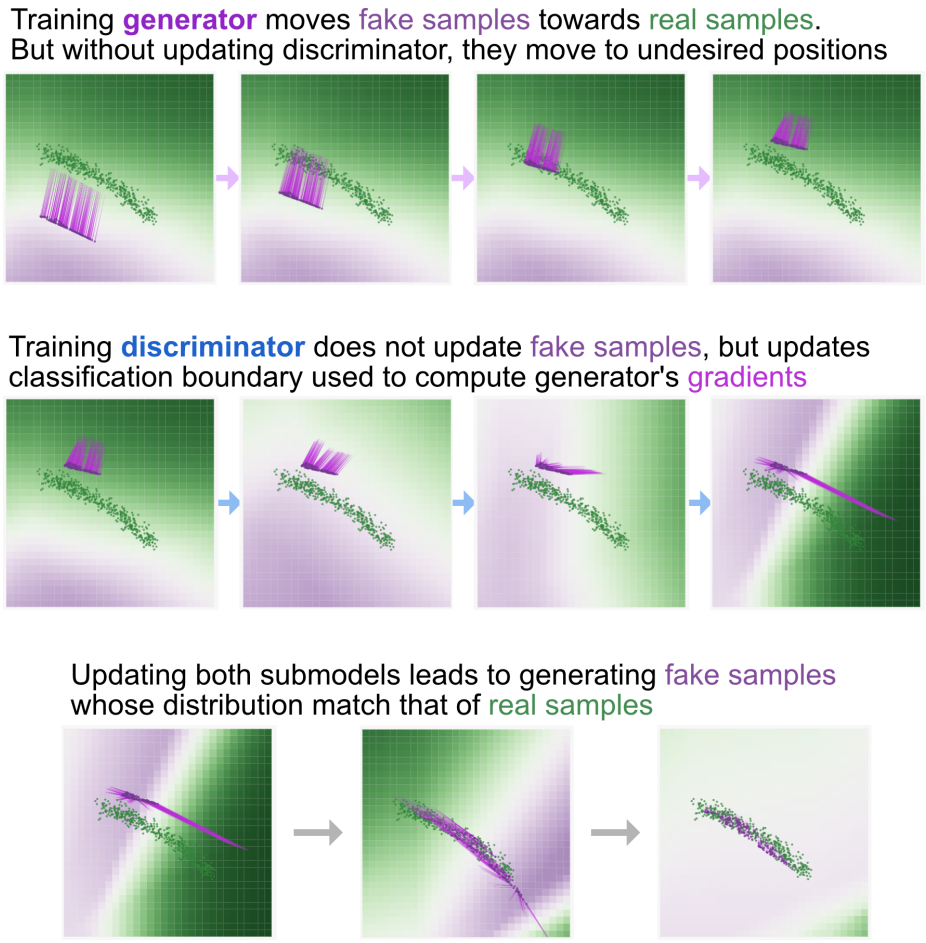


Figure 6.13: Experimenting with manual step execution, to understand the interplay between discriminator and generator.

For example, she is able to find that while discriminator's loss is computed by using both real and fake samples, only fake samples are used when computing the generator's loss.

Understanding the different roles of discriminator and generator with the manual step execution. While the slow-motion mode has helped her better understand the steps of the training process, Alice wonders how the discriminator and generator play a "game" to generate data distributions. To analyze the different effects for the discriminator and the generator, she would like to experiment with the two submodels using the manual step-by-step execution feature. She clicks the button to update the generator. Her initial clicks cause the fake samples to move towards the real samples, but as she clicks a few more times, the

fake samples "overshoot," no longer matching real samples' distribution (Figure 6.13, top row). She now realizes that the fake samples have moved towards regions where the colors of background grid cells are green, not directly towards the real samples. This leads Alice to hypothesize that training the discriminator is necessary for the generator to produce better fake samples. So, she switches to only training the discriminator, which does not reposition the fake samples, but the grid colors update (Figure 6.13, second row) to correct a decision boundary that separates the real and fake samples. She believes that this new boundary helps guide the fake samples towards desirable regions where the real samples are located. This experiment helps her realize that updating both submodels is important for generation of better fake samples. Now she clicks the buttons for updating the discriminator and generator alternatively, which successfully creates a fake distribution that matches the real distribution. That is, the discriminator cannot distinguish between real and fake samples. (Figure 6.13, last row).

6.8.2 Practitioners Experimenting with Hyperparameters

One of *GAN Lab*'s key features is the interactive, dynamic training of GANs. Experimentation using *GAN Lab* could provide valuable practical experience in training GAN models even to experts. Consider Bob, a machine learning engineer at a technology company.

Guiding models to find the optimum. Bob launches *GAN Lab* and starts the training process. Fake samples quickly move towards real samples. However, as the training progresses, he notices that the fake samples oscillate around the real samples. Based on his previous experience, he believes this indicates that the *learning rates* may be set too high. He first decreases the value for the discriminator by using the dropdown menu, but the amount of oscillation becomes more severe. By checking the interface, he quickly realizes that there are two learning rates in GANs, so he reverts its value and decreases the generator's learning rate. After a few more iterations, the oscillation subsides and the distribution of the fake samples almost matches that for the real samples. This experimentation

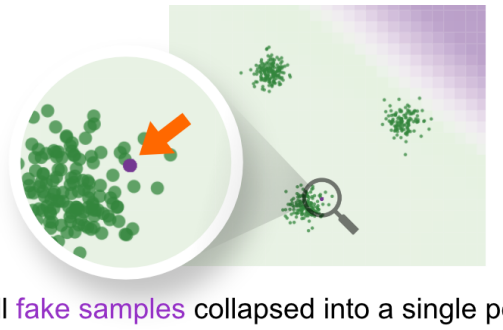


Figure 6.14: Mode collapse, a common problem in GANs.

helps him understand the importance in balancing the power between the discriminator and generator.

Understanding equilibrium between discriminator and generator. Bob wonders what would happen if he perturbs the equilibrium between the discriminator and generator. That is, what if either submodel overpowers its complement. Looking into the model overview graph, he finds that some other hyperparameters also come in matched pairs, such as the number of training loops, one for the discriminator and the other for the generator. Originally, both numbers are set to 1 (i.e., the submodels run one training epoch in alternate sequence). Bob decides to increase discriminator's loop count 3 (i.e., 3 discriminator epochs, followed by 1 generator epoch, followed, and repeat). To his surprise, this "unbalanced" epoch setting (3 vs. 1) causes GAN to converge faster. Comparing this "unbalanced" setting with the original "balanced" (1 vs. 1) setting, Bob starts to understand that a more powerful discriminator can indeed accelerate training, because a stronger discriminator leads to stronger gradients for the generator, which in turns more quickly move the fake samples towards the real distribution, thus faster training convergence.

Exploring mode collapse. Bob would like to train a GAN to work with more complex data distributions. He picks one distribution that consists of three disjoint dense regions. He increases the number of layers for both the generator and discriminator, then clicks the play button. After a few seconds, all fake samples seem to have disappeared, as he can

only see real samples. He temporarily hides the real samples (by toggling their visibility), thinking that they may be covering the fake samples. Then, he realizes that all fake samples have collapsed into a single point (as shown in Figure 6.14). He does not know why this happens, and wonders if it is due to his hyperparameter choices. So he experiments with several other sets of hyperparameters, and observes the pattern that this happens more often when the generators and discriminators are set to use more layers and neurons. He consults the literature for possible causes, and learns that this is in fact a well-known problem in GANs, called *mode collapse*, whose exact cause is still an active research topic [66, 127]. Bob's observation through *GAN Lab* motivates him to study new variants of GANs, which may overcome this problem [66, 127].

6.9 Observational Study

To investigate how *GAN Lab*'s target users (e.g., students aspired to learn about GANs) would use the tool and learn about the models, we conducted a small observational study. This section describes our study design and findings.

6.9.1 Study Design

Participants. Six participants were recruited through our institution's mailing list for those who are interested in machine learning. We pre-screened participants to ensure that they have at least basic knowledge of deep learning and GANs (e.g., taken a deep learning course or at least heard of GANs). Five participants were Ph.D. students who had taken a deep learning course, and one was an undergraduate student who had research experience. They self-reported their level of knowledge on deep learning, with an average score of 3.3 on a scale of 0 to 5 (0 being "no knowledge" and 5 being "expert"); and that on GANs with

[5]This section is adapted from work appeared at EVIVA-ML Workshop at IEEE VIS 2019 [87].

Table 6.1: Subjective ratings about *GAN Lab* using 7-point Likert scales (7: *Strongly Agreed*. 1: *Strongly Disagreed*).

Question	**Avg.**
Easy to learn how to use	6.3
Easy to use	6.3
Helpful to understand what constitutes a GAN model	6.5
Helpful to understand the training process of GANs	6.0
Helpful to understand what the generator is doing	5.5
Helpful to understand what the discriminator is doing	6.2
Helpful to understand how hyperparameters affect results	6.2
Helpful to get new insight about GANs	5.8
I felt confident when using the tool	5.8
It improves the effectiveness of my learning	5.7
I enjoyed using *GAN Lab*	6.5
I would like to use software like *GAN Lab* to learn ML	6.5

an average score of 2.5 (on the same scale). No participant has used or heard about *GAN Lab* before.

Procedure. The study was conducted through BlueJeans video conferencing. After the participants signed their consent forms electronically, they were provided a 5-minute overview of GANs, followed by a 5-minute tutorial of *GAN Lab*, which described its visualizations and features. After that, the participants freely explored using *GAN Lab* on their computer's web browser. They were asked to think aloud and share their computer screen with us during the study. They could ask for questions when necessary. After they used the tool, the participants were asked to fill out questionnaires, consisting of subjective ratings about *GAN Lab* (12 questions) and questions for feedback (4 questions). The study took about 50 minutes, and each participant was compensated with an Amazon $15 gift card for their time.

6.9.2 Questionnaire Results

Subjective ratings. We measured several aspects of *GAN Lab* using 7-point Likert scales (7 being *Strongly Agreed*; 1 being *Strongly Disagreed*). Table 6.1 shows the average ratings for the 12 questions we asked. The participants found that *GAN Lab* was easy to learn, easy to use, helpful to understand several aspects of GANs, and likeable overall. Specifically, all six participants found *GAN Lab* easy to learn to use (i.e., rated 6 or 7), and all but one participant agreed that *GAN Lab* was easy to use, they enjoyed using it, and they would like to use software like to learn machine learning. Five questions starting with "helpful to understand" are asking whether *GAN Lab* improves their understanding of certain aspects of GANs. The question that received the highest average rating was on what a GAN model is composed of, which indicates that *GAN Lab*'s visualization was effective. In addition, the only question that all participants agreed was on the understanding of the effects of hyperparameters, related to the *GAN Lab*'s interactive experimentation features. Even with a variety of features that aim to improve the understanding of the generator, participants reported that it was relatively harder to understand the generator, in terms of the average rating (i.e., 5.5), while the value is high enough to say it is positive.

Qualitative feedback. We asked participants for feedback on *GAN Lab*. Participants liked a variety of visualizations and features it provided. For example, multiple participants said they liked *GAN Lab*'s visualizations that evolve as the training process progresses. One participant said "*I liked the updated visualizations of the manifold, gradients, etc. I liked these because it provided insight as to how the GAN was evolving in time, which provides insight into how it works and what the end goal of a GAN is.*" Another said "*I did learn more properly how the GANs actually evolve, as I did not fully understand how they operated before. I don't think my DL professor explained as nicely as how this tool demonstrated.* " In addition, multiple Participants particularly liked the feature for adjusting hyperparameters. We report them and other feedback more in detail in the next

subsection.

6.9.3 Key Findings

Rapid hypothesis testing. Among the features of *GAN Lab*, many participants particularly liked the one for dynamically adjusting hyperparameters while a model was being trained. This feature enabled them to form hypotheses based on prior experience in machine learning and rapidly test them using *GAN Lab*. For example, one participant increased the learning rate (using its drop-down menu) to test if it helps speed up the training. Another participant said "*I really liked the features of the hyperparameter tuning [...], and learning all the different hyperparameters that can affect them are making me think of different ways to optimize GANs.*" This capability for rapid hypothesis testing in *GAN Lab* is not possible in conventional deep learning workflows because they often require retraining the model each time a user adjusts a hyperparameter.

Building intuition through dynamic experiments. The ability to adjust hyperparameters in *GAN Lab* also helps users build intuition about the behaviors induced by the model's training process. One important characteristic of GANs is the dynamic interplay between the two components: generators and discriminators. A participant said "*[the] ability to change training parameters such as number of updates on the fly was nice. It really helps you build intuition to see how the discriminator and generator interact.*" One usage pattern participants particularly liked was updating either the generator or discriminator while disabling the update of the other. By default, the training process alternates between the generator and discriminator (in each iteration), so it can be hard for novices to understand their individual contribution to the training progress. By disabling one of them, users can more easily observe how each component works and how the model reaches an equilibrium that balances the two components.

Validating knowledge from literature. Participants who are familiar with the literature of deep learning and GANs found *GAN Lab* useful for validating knowledge they ac-

quired from research articles. For example, one participant remembered that GANs would often encounter the problem called *mode collapse*, especially when a distribution contained disjoint modes [127]. This participant was interested in reproducing this phenomenon by training a model with such a distribution. He also wanted to use a different loss function that might mitigate this issue, as suggested in the literature. This observation suggests that interactive tools like *GAN Lab* may help not only novices learn the basic concepts of models, but also researchers and practitioners validate knowledge they learned from the literature, which could help them build trust in the model's training process.

Beginners need further guidance. We observed that participants less familiar with GANs needed more guidance to help them fully enjoy the tool. Some were not sure about what to try. One said "*helpful to [provide descriptions] of what GANs training scheme "works" and what "doesn't work.*"" Although we wanted users to self-discover relationships between hyperparameters and results by actively playing with the tool, it might be beneficial for us to also provide step-by-step exercises that would guide users' experimentation, similar to how TensorFlow Playground has been integrated into Google's machine learning course material on the web [150]. The course includes a series of exercises which learners can follow. For example, in the chapter on learning rates, learners are asked to try different learning rates and compare the results.

6.9.4 Discussion: Measuring Understanding Level

Our observational study is an early step in understanding how people may learn deep learning through interactive education tools. There remain many challenges in designing controlled experiments to further such evaluation efforts. One important challenge is the choice of dependent variables that measure a user's level of the understanding in machine learning models, similar to the use of task completion time for evaluating information exploration tools. We briefly discuss this challenge here.

Studies conducted in computer science education research and those for evaluating al-

gorithm visualizations (in early 2000s) typically included pre- and post-study tests that sought to measure participants' conceptual or procedural knowledge (e.g., what is the algorithm's time complexity, what would be the next state after '17' is inserted) [79]. However, test questions suitable for simpler, deterministic algorithms may not generalize to modern machine learning models that are often complex and probabilistic.

Thus, it would be a valuable effort to develop new ways to evaluate the educational effectiveness of interactive tools for machine learning. Below we present a few ideas. First, the computer science education literature has developed several methods, such as analyzing mental models or measuring self-efficacy [133, 146], and we can draw inspirations from them. Next, inspired by how visual analytics tools are evaluated [135], studies may be designed to analyze if participants discovered new insights on machine learning models. In addition, since the primary goal of ML learners is often in developing models for real data, it could be helpful to design studies that assess if users are able to implement models with high accuracy.

Lastly, we wanted to note that the level of understanding is not the only dependent variable in evaluating educational tools. Another important factor to measure is the learners' engagement level [134]. A high level of engagement (e.g., spending more time and efforts) often indicates that users enjoy the tool and may likely learn more through the usage. To investigate if *GAN Lab* users are actively engaged, in the next section, we describe our analysis of anonymous usage log (e.g., buttons users clicked) from our deployed website.

6.10 Log Analysis of Deployed Tool

As mentioned earlier, we have deployed *GAN Lab* on the web at `https://poloclub.github.io/ganlab/`. Users can play with *GAN Lab* using their browsers. This website also contains a short introduction of GANs and a tutorial for the tool, which can be found at the bottom of the tool. Since launched in September 2018, it has received significant

attention. Within the first year, more than 70,000 people from over 160 countries tried it out (according to Google Analytics).

To investigate whether users of the deployed tool are engaged with *GAN Lab* by using a variety of features it provides, we conducted a study on an analysis of users' interaction log. Analyses of users' interaction histories have been widely used to evaluate visual analytics tools [145, 53, 68]. Previous studies demonstrated that careful examinations of a user's interaction log can recover the user's reasoning process [53]. Both automated techniques and manual reviewing have been used [68]. We use a semi-automated approach that first manually identifies common *actions* and automatically extracts the identified actions from logs.

6.10.1 Data Collection

We have collected anonymous interaction logs from the deployed website. The logs mainly include users' clicks on HTML elements. We analyze five weeks of data collected from July 27 to August 30, 2019. We did not collect data from users located in European countries determined based on their computers' timezone information because of IRB-related issues, and we also did not use data for users who opted out by clicking the corresponding link on the website. The study has been approved by Georgia Tech's IRB, and consent forms were waived. The collected data are stored in databases on Google Cloud.

Summary statistics. The collected data contains 39,705 click events by 2,218 users (17.9 clicks on average). Among 2218 users, 950 users clicked the elements on *GAN Lab* at least 10 times, 330 users at least 30 times, and 59 users at least 100 times. To analyze the behavior of users who had sufficiently interacted with the tool, we decided to analyze the interaction logs for the 330 users who clicked the elements at least 30 times. An average click count by these 330 users is 73.9.

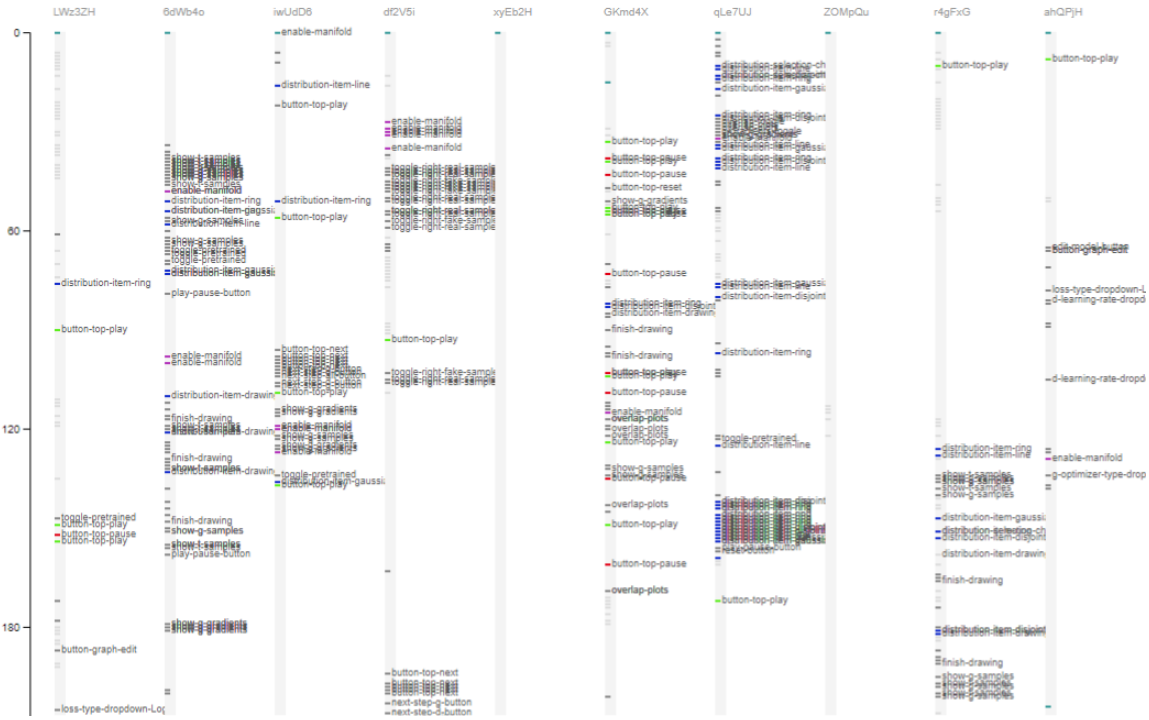


Figure 6.15: Screenshot of a visualization tool of usage log, which we developed for exploring and identifying common actions. Each column represents a sequence for a single user. The vertical axis represents the number of seconds since a user visited *GAN Lab* the first time. Clicked HTML elements are shown as a small rectangle on the vertical bar for each user's sequence, with a label.

6.10.2 Exploring Data and Identifying Actions

There are 91 different HTML elements clicked by at least one of the all 2,218 users. The elements range from the play button to several drop-down menus. For example, the most popular element was the play button located on the top of the interface, which was clicked at least one time by 327 users among the 330 users.

While performing data analysis at an element-level provides a basic information of usage statistics, we are interested in higher-level semantically meaningful behaviors of users. Thus, we decided to identify a list of common *actions*, similar to Gotz and Zhou [68]'s *action* tier, a richer level of semantics not found in lowest-level user interaction event (e.g., mouse click). We first went over a sample of user logs to identify common actions. To do

that, we developed a visualization tool that lists event sequences of users (as shown in Figure 6.15, similar to that used in the literature [53]. After the exploration of the sequences, we identified the following 9 common actions:

1. Select a different pre-defined data distribution and train a model
2. Draw a distribution and train a model
3. Enable and inspect the generator's manifold visualization
4. Train a model in a submodel-level (either only a discriminator or generator)
5. Enable and inspect the training process using the slow-motion mode
6. Change the size of submodels (e.g., layers, neurons)
7. Adjust hyperparameter(s) (e.g., learning rates) while a model is being trained
8. Change the number of training iterations for submodels
9. Read instructions located under the tool

For each of the 9 actions identified, we have written a script that finds matching patterns from the logs. For example, to determine whether a user had adjusted hyperparameters while a model was being trained, the script first selects users who clicked corresponding HTML elements (e.g., item in the dropdown menu for learning rates) and checks if the iteration count had been increased after the click event. We have iteratively refined the script by incrementally adding constraints, to accurately reflect the identified actions. For instance, to determine if a user had used the slow-motion mode, we first simply checked if they clicked the button for the slow-motion mode, however, we soon realized that some users clicked the same button right after their first click, which means they unlikely used the feature, so we have revised the script to count users only when they used the feature at least for 10 seconds.

Table 6.2: Numbers of users who performed each of the 9 common actions identified from their click logs (among 330 users who clicked HTML elements at least 30 times)

Action	**# of Users**
1. Select a different distribution	303
2. Draw a distribution	207
3. Train in a submodel-level	88
4. Use slow-motion	99
5. Change submodel size	126
6. Adjust hyperparameters during training	99
7. Adjust # of training iterations for submodels	56
8. Enable and inspect generator visualization	213
9. Read instructions	135

6.10.3 Results

Table 6.2 shows the number of users (among the 330 users) who performed each of the 9 actions. For example, the second row indicates that 207 users (63% out of 330) drew at least one data distribution by themselves using the *GAN Lab*'s feature for drawing new distributions and trained a GAN model for the distribution. The results demonstrate that many of users were able to play with *GAN Lab* by using a variety of features, even though all these users are anonymous users who visited our website voluntarily. For instance, a large number of users trained GAN models by selecting multiple different data distributions available on the interface (i.e., #1, #2). In addition, many users investigated the interplay between the two submodels, the generator and discriminator, by adjusting a parameter for one of them (e.g., train either a generator or discriminator in #3). Furthermore, many visitors directly manipulated a range of hyperparameters (i.e., #5, #6, #7). In sum, we are excited that users were able to enjoy a variety of features provided by *GAN Lab*, only with short tutorials provided in the demo page.

6.11 Limitations and Future Work

Transferring user knowledge to higher dimensions. Our main decision to use 2D datasets is to promote comprehension [156]. Through our tool, with 2D datasets, users can gain important knowledge about the overall training process of GANs, and specific details, such as how model components interact over time, how data flow through components, and how losses are recomputed to iteratively update components. These important concepts and knowledge are transferable to practical use cases of GANs where higher dimensional data are used (e.g., images). However, it remains an open research problem whether certain behaviors (e.g., mode collapse) that users may observe when experimenting with 2D datasets would be easily reproducible in higher dimensional datasets, where the larger number of parameters would lead to more-complex interactions and less-predictable results. We plan to conduct studies to develop deeper understanding of how and when such correspondence or mismatch may occur.

Supporting image data. To extend *GAN Lab* to support image data, some modifications and optimizations will be needed. Training on image data is often time consuming. To speed this up, pre-trained models may be provided to users so they can skip the earlier training steps. As for visual design, projection methods (e.g., t-SNE) may be used to replace some views in *GAN Lab* to visualize the distribution of generated image samples [177].

Speed and scalability. *GAN Lab* leverages TensorFlow.js to accelerate GAN training for browser-based deployment. For models with many parameters, this can be time consuming. In the short term, we believe rapid advances in JavaScript and hardware will shorten this by a good amount. A longer-term challenge to overcome is browsers' inability to render visualization and perform computation at the same time (i.e., single-threaded). Developers need to strike a good balance in planning and interleaving these actions, to maximize model computation speed and visual responsiveness.

Supporting more GAN variants. While *GAN Lab* currently implements a few differ-

ent loss functions, other GAN variants exist [75]. Through open-sourcing *GAN Lab*, we look forward to seeing the community to build on *GAN Lab* to implement more variants, enabling users to interactively and visually compare them, easing the challenges in evaluating GANs [66]. Some variants may require minor design changes of the interface (e.g., adding new nodes to overview graph).

In-depth evaluation of educational benefits. Longitudinal studies of *GAN Lab* will help us better understand how it helps with learning of GANs. It would be particularly valuable to investigate how different types of users (e.g., students, practitioners, and researchers) would benefit from the tool.

CHAPTER 7

ETABLE: INTERACTIVE BROWSING AND QUERYING OF RELATIONAL DATABASES

One of the first steps in machine learning is making sense of raw datasets. While many different types of databases and storage systems exist, relational databases are still one of the popular databases used in the enterprise. Researchers have devoted considerable attention to helping database users formulate queries, however, many users find it challenging to specify queries that involve joining tables in relational databases. To help users construct join queries for exploring relational databases, this chapter presents *ETable*, a novel presentation data model that provides users with a presentation-level interactive view. This view compactly presents one-to-many and many-to-many relationships within a single *enriched table* by allowing a cell to contain a set of *entity references*. Users can directly interact with this enriched table to incrementally construct complex queries and navigate databases on a conceptual entity-relationship level. In a user study, participants performed a range of database querying tasks faster with *ETable* than with a commercial graphical query builder.

7.1 Introduction

A considerable challenge for non-technical users of relational databases is constructing *join* queries [81]. The join operation is required for even simple data lookup queries since

This chapter is adapted from work appeared at VLDB 2016 [89].

Papers filtered by `Paper_keywords.keyword like '%user%' AND Conferences.acronym = 'sigmod'`

id	title	year	page_…	page_…	Conference acrony	Authors names	Papers (referencing) titles	Papers (referenced) titles	Paper_keywo keywords
2575	Making database systems usable	2007	13	24	SIGMOD	H. V. Jaga…, Adriane Ch…, Aaron Elki…, Magesh Jay…, Yunyao Li 7	XRANK: Ran…, NaLIX: an…, DaNaLIX: a…, Assisted q…, Towards a… 12	QueryViz:…, Exploring…, Efficient…, Homebrew d…, The intera… 25	user inter fact…, gen usability
2628	Addressing diverse user prefer…	2007	641	652	SIGMOD	Zhiyuan Ch…, Tao Li 2	Adaptive w…, Enhanced w…, Context-se…, Automatic…, Ordering t… 10	Making dat…, Supporting…, Skimmer: r…, Diversity…, Efficient… 13	informatio prefe…, da human fact algorithms
2701	Assisted querying using instan…	2007	1156	1158	SIGMOD	Arnab Nand…, H. V. Jaga… 2		Predicting…, The intera…, FreeQ: an…, Efficient…, Location-a… 8	query, key interface, autocomple inter…

Figure 7.1: *ETable* integrates multiple relations into a single enriched table that helps users browse databases and interactively specify operators for building complex queries. This example presents a list of SIGMOD papers containing the keyword "user" from an academic paper database collected from DBLP and the ACM Digital Library. Each column represents either a base attribute of a paper or a set of relevant entities obtained from other tables (e.g., `Conferences`, `Authors`). If a relational database were used to obtain the same information, 9 tables would need to be joined, and the results produced can be hard to interpret because of many duplicated cells.

relational databases store information in multiple separate normalized tables. Although database normalization provides many benefits for managing data (e.g., avoiding update anomalies), it significantly decreases the usability of database systems by forcing users to write many join queries to explore databases.

Constructing join queries is difficult for several reasons. The main reason is that users find it difficult to determine which relations to join among many relations. Understanding the role of each relation that represents a relationship of interest and finding the right join attributes are not trivial tasks, even when a schema diagram is given. To tackle this challenge, users often write complex queries by starting with a simpler query and iteratively adding operators [130]. Although this iterative strategy is helpful, it is still challenging because the format of join query results is hard to interpret. For example, consider a query that joins two relations in many-to-many relationships (e.g., `Papers` and `Authors` in Figure 7.3). A result of this query produces a large number of duplications (e.g., the title of

each paper repeated as many times as the number of its authors). People represent the same information differently when they use a spreadsheet. For instance, they might create a cell containing multiple values separated by commas. Relational databases cannot represent data in this way because the relational model (as implemented in most relational DBMSs) requires that data be at least in the *first normal form*.

The usability challenge of writing complex queries has been studied by many researchers. Although *visual query builders* help people formulate SQL queries [33], they separate query construction and result presentation parts [81], introducing a usability gap between users' actions and their results [160, 130]. To overcome this limitation, researchers argue that database interfaces need to adopt the *direct manipulation* principle [160], a well-known concept in the *human-computer interaction (HCI)* area [81, 116]. It enables users to iteratively specify operators by directly interacting with result instances using simple interactions [116]. Researchers also argue that join query results should be represented in an easier-to-understand format that improves the interpretation of query results. Jagadish et al. [82] proposed the notion of the *presentation data model*, which they defined as a full-fledged layer above the logical and physical schema. This presentation layer allows users to better understand the query results without requiring full awareness of the schema. All this research strongly suggests the need for developing database interfaces that are usable, interactive, and interpretable.

We present *ETable*, a novel presentation data model with which users can interactively browse and navigate databases on an entity-relationship level without writing SQL. *ETable* presents a query result as an enriched table in which each cell can contain a set of *entity references*. By deliberately relaxing the *first normal form*, we compactly represent one-to-many and many-to-many relationships within a single table — a novel capability that enables users to more easily browse and interpret query results consisting of multiple relations. Figure 7.1 illustrates how *ETable* effectively presents a list of SIGMOD papers containing the keyword "user" from an academic paper database collected from DBLP and

the ACM Digital Library (see Figure 7.3 for schema). Each row in *ETable* shows the base attributes and relevant entities of a paper, such as its authors and cited papers. If a relational database were used to obtain the same information, 9 tables would need to be joined, and the results produced would be hard to interpret (e.g., many duplicated cells).

To discover which relevant entities should be shown for each row, *ETable* uses a novel graph-based model called the *typed graph model (TGM)*, which frees users from concerning themselves with the complexity of the logical schema; users may instead focus on exploring and understanding the dataset at the conceptual (or entity-relationship) level. The *typed graph model* stores relational data as graphs in which nodes represent entities (e.g., authors, papers) and edges represent relationships (e.g., those that relate authors to papers). This transformation enables *ETable* to retrieve other related entities through simple graph operations. For example, a given paper's authors, stored as direct neighbors, can be retrieved through a quick neighbor-lookup.

As the construction of complex queries and the exploration of data are inherently iterative processes, database exploration tools should provide easy-to-use operations to help users incrementally revise queries [36, 130, 116]. *ETable*'s direct manipulation interface enables users to directly work with and modify an existing enriched table to update its associated queries. For example, imagine a user, Jane, who would like to further explore the result in Figure 7.1. To see the detailed information about the authors of a particular paper, she clicks on its "author count" button (Figure 7.2-b). This simple interaction of tapping the button is translated into a series of *primitive operators* behind the scene, such as *Select*, as in selecting the row associated with a paper; and *Add*, as in adding and joining the `Authors` table with the `Papers` table. With a few rounds of similar interactions, Jane can incrementally build complex queries.

ETable's novel ideas work together to address an important, often overlooked problem in databases. The seminal vision paper by Jagadish et al. [81] introduced the notion of the presentation data model and argued the importance of direct manipulation interface.

However, designing an easy-to-use system that meets these requirements is challenging. *ETable* is one of the first instantiations of this important idea, filling a critical research gap, by effectively integrating HCI principles to greatly improve database usability. To enable the creation of such a usable tool, *ETable* tightly integrates: (1) a novel hybrid data model representation, which advances over the relational and nested-relational models, to naturally represent entities and relationships; and (2) a novel set of interactions that closely work with the representation to enable users to specify expressive queries through direct manipulation. With *ETable*'s user interface, non-experts can easily and naturally explore databases without writing SQL, while *ETable* internally performs queries under the hood.

Through *ETable*, we contribute:

- A novel **presentation data model** that presents a query result as an enriched table for users to easily browse and explore relational databases (Section 7.3, 7.5);
- A **graph-based model**, called *typed graph model (TGM)* that provides an abstraction of relational databases, for users to explore data in *ETable* at a conceptual level (Section 7.4);
- A set of **user-level actions**, operations that users can directly apply to an enriched table to incrementally construct complex queries and navigate databases (Section 7.6.1);
- The **usable interface** of *ETable* that outperforms a commercial graphical query builder in a **user study**, in both speed and subjective ratings across a range of database querying tasks (Section 7.6, 7.7).

7.2 Related Work

Database Usability and Query Specifications. Since Query-by-Example (QBE) was developed in 1970s [193], database researchers have studied fairly extensively the usability

aspect of database systems [81, 32, 4, 80]. Usability is important, especially because not all database users have expertise in writing complex queries; many non-technical users find it challenging to write even very simple join queries [81, 1]. Many existing approaches are aimed at assisting users with formulating queries. One representative method is the *visual query builder*, which enables users to visually manipulate schema elements on a graphical interface [33]. However, most visual querying systems require that users have precise knowledge of a schema, which makes it difficult for non-experts to use. This limitation can be relieved in *keyword search* systems, studied extensively in the last decade [78, 25, 6, 41], or natural language interfaces [109]. However, most of existing approaches [83, 59] separate queries and results so that users cannot directly refine query results, which decreases the usability of the systems. Nandi and Jagadish [130] argued that users' querying process is often iterative, so database systems should guide users toward interactively formulating and refining queries.

Direct Manipulation and Iterative Querying. Several database researchers argued that the usability of database querying systems can improve by adopting the *direct manipulation* paradigm [160], a well-established design principle in the HCI and information visualization areas. Acknowledging that users' needs are often ambiguous rather than precisely specifiable, researchers have developed many tools that enable users to interactively browse and explore databases [80, 29, 161]. Although they are not specifically designed for relational databases, a number of interactive visualization systems for entity-relationship data have been developed by information visualization researchers [92, 56, 51, 121]. For example, NetLens [92] visualizes relationships between two selected entity types in many-to-many relationships, and GraphTrail [56] visually summarizes each entity type and enables users to switch between entities. Although these visualization systems provide an overview of datasets, they are not suited for examining database instances along with attributes. In exploring and analyzing instance-level information, tabular interfaces, including spreadsheets, are better suited and often preferred by database users [60, 172, 116, 37,

65]. Tyszkiewicz [172] argued that spreadsheets can play a role as a database engine by using functions and macros. Liu and Jagadish [116] formally defined operators that interactively perform grouping operations within a spreadsheet. However, since the rigid tabular structure does not effectively present many-to-many relationships, the spreadsheet suffers from the same problems that relational databases have (i.e., a large number of duplications). To overcome this limitation, Jagadish et al. [82] proposed using a presentation view layer on top of underlying databases, which is the notion of the *presentation data model*, defined as a full-fledged layer on top of the logical and physical models. The challenge is to design presentation data models that help people easily understand join query results and interact with them.

Data Models for Effective Presentation. To develop an intuitive structure for presentation data models, we review a number of data models that conceptualize the mini-world represented in databases. One such example is the *nested relational model*, studied in the 1980s, which allows each cell to contain another table that presents one-to-many relationships in a single table [155, 151]. The nested model has been used in several studies for designing database interfaces. Bakke et al. [19] designed a direct manipulation interface for nested-relational databases, and DataPlay [5] also used the nested model for presenting query results. However, the model suffers from scalability issues because the sizes of the nested tables often become huge when an inner table contains a large number of associated rows or columns [20]. One way to tackle this problem is to replace the inner table with a set of pointers. For example, the *object-relational model* lets attributes be user-defined types that include pointers [166]. We adapt this idea by introducing an *entity reference* which compactly represents related entities. Another class of the data models that effectively conceptualize the real-world is the graph data model [13, 70, 34, 169]. It represents entities as nodes and relationships as edges based on the *entity-relationship model* [40, 23]. Catarci et al., [35] used a graph-style *translation layer* for their visual querying system. To provide users with an easy-to-understand view at an entity-relationship level, we also maintain a

graph-style model, transformed from relational databases, under the presentation view.

7.3 Introducing *ETable*

Before we describe the technical details of the proposed data models, we introduce *ETable* by describing what users see and how they can interact with it.

Representation. Figure 7.1 illustrates an enriched table that we call *Etable*. As mentioned earlier, it presents a list of SIGMOD papers containing the keyword "user" from our collected database (see Figure 7.3 for schema). Each row of *Etable* represents a single entity of the selected entity type (i.e., `Papers`); its column represents either a base attribute of the entity (e.g., year) or a set of relevant entities (e.g., authors, keywords). This representation is formed by pivoting a query result of a join of multiple tables (e.g., `Papers`, `Paper_keywords`, `Authors`) to a user-selected entity type (e.g., `Papers`). One advantage of this representation is that it can simultaneously present all relevant information about an entity in a single row (e.g., authors, keywords, citations). The relational model cannot represent all of this information in a single relation without duplications because every attribute value must be atomic. For instance, when the `Papers` table is joined with the `Authors` table, the paper information is repeated as many times as the number of authors, which prevents users from quickly interpreting the results. We integrate information spread across multiple tables into a single table by allowing each cell to contain a set of references to other entities.

Interactions. Users can interact with *Etable* to explore further information. For instance, to examine further information about the authors of the papers in Figure 7.1, users can create a new *Etable* that lists authors in several ways, as depicted in Figure 7.2: (1) If users are interested in one of the authors (e.g., Arnab Nandi), they can click on his name to create a new *Etable* consisting of one row that presents its attributes; (2) if users want to list the complete set of authors (e.g., all seven authors of the paper titled "Making database

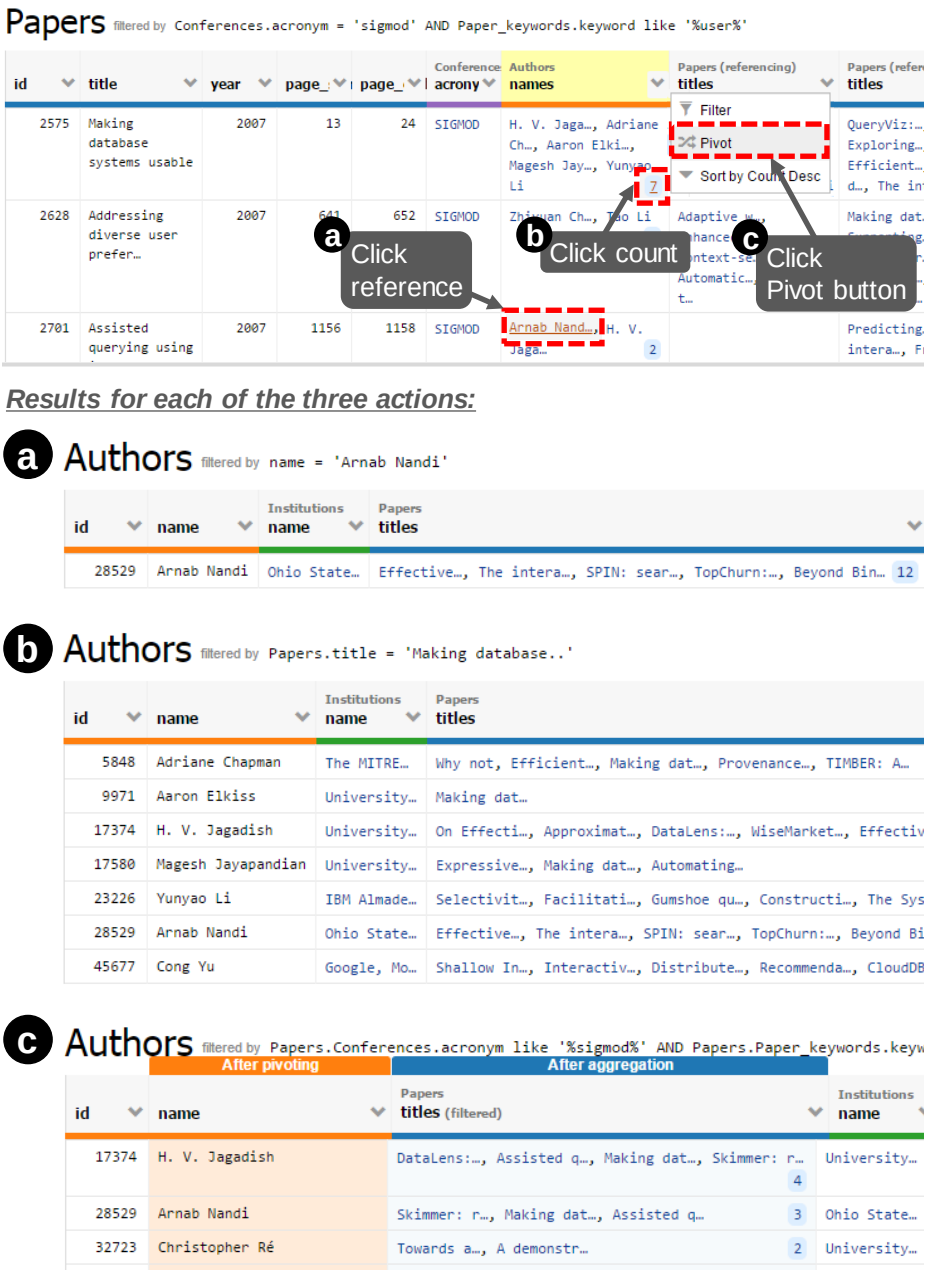


Figure 7.2: Users can iteratively specify user-level actions by interacting with *ETable*. In this example, users can examine further information about paper authors in three ways: (a) clicking on an author's name; (b) clicking a paper's author count; (c) clicking on the pivot button.

systems usable"), they can click on the author count in the right corner of the cell (i.e., 7); and (3) if users want to list and sort the entities across the entire rows in a column (e.g.,

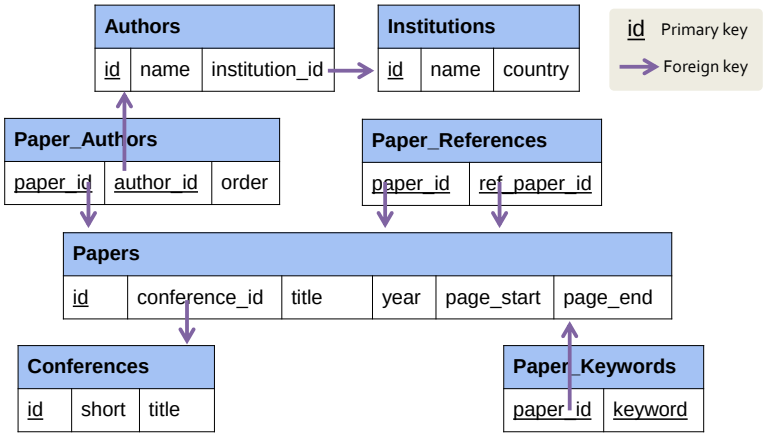


Figure 7.3: The relational schema of the academic dataset used in this work, 7 relations in total.

Who wrote the most papers about "user" in SIGMOD?), they can click on the *pivot* button on the column menu, which groups and sorts the authors based on the number of papers they have written. By gradually applying these operations, users can incrementally make sense of data and build complex queries.

7.4 Typed Graph Model

In this section, we define a *typed graph model (TGM)* which enables users to explore relational databases on a conceptual entity-relationship level without having to know a logical schema. A relational schema and instances are translated into a *database schema graph* and *database instance graph* as a preprocessing step, and all operations specified by users on the *ETable* interface are executed over these graphs, not relational databases.

We represent entities and relationships as a graph with types and attributes. Each entity forms a node, and relationships among the entities become edges. A *typed graph database (TGDB)* consists of a *TGDB schema graph*, $\mathcal{G}_S$, and a *TGDB instance graph*, $\mathcal{G}_I$.

Definition 1. Schema Graph. A TGDB schema graph $\mathcal{G}_S$ is a tuple $(\mathcal{T}, \mathcal{P})$, where $\mathcal{T}$

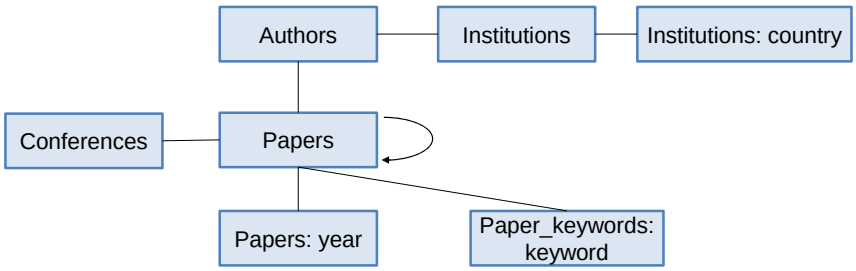


Figure 7.4: TGDB schema graph constructed from the relational schema in Figure 7.3. Each rectangle represents a node type, and each edge is an edge type.

represents a set of node types (or entity types[1]), and $\mathcal{P} \subseteq \mathcal{T} \times \mathcal{T}$ represents a set of edge types (or relationship types). Each node type $\tau_i \in \mathcal{T}$ is a tuple $(\alpha_i, \mathcal{A}_i, \beta_i)$, where α_i denotes the name of a node type, $\mathcal{A}_i$ is a set of single-valued attributes, and β_i is a label attribute chosen from one of the attributes and used to represent node instances of this type. Each edge type $\rho \in \mathcal{P}$ also has a name and a set of attributes. We denote the source and target node types of ρ as $source(\rho)$ and $target(\rho)$, respectively. All the edge types, except self loops, are bidirectional.

Definition 2. Instance Graph. A TGDB instance graph $\mathcal{G}_I$, is a tuple (V, E), where V represents a set of nodes (or entities) and E represents a set of edges (or relationships) between two nodes. Every instance graph $\mathcal{G}_I$ has a corresponding schema graph $\mathcal{G}_S$, and the instance graph has a node type mapping function $type_\tau : V \rightarrow \mathcal{T}$ and an edge type mapping function $type_\rho : E \rightarrow \mathcal{P}$ that partition nodes V into $V_1, ..., V_{n_\mathcal{T}}$ and edges E into $E_1, ..., E_{n_\mathcal{P}}$. Each node $v \in V$ consists of a set of attribute values $v[A_{ij}]$ for the attributes of the corresponding node type and has a label defined as $label(v) = v[\beta_i]$. Each edge $e \in E$ consists of a set of attribute values $e[A_{ij}]$ for its type. We denote the source and target nodes of e as $source(e)$ and $target(e)$, respectively.

[1]We use the words "node" and "entity" interchangeably. A node is used more formally; an entity is used more for presentation to users.

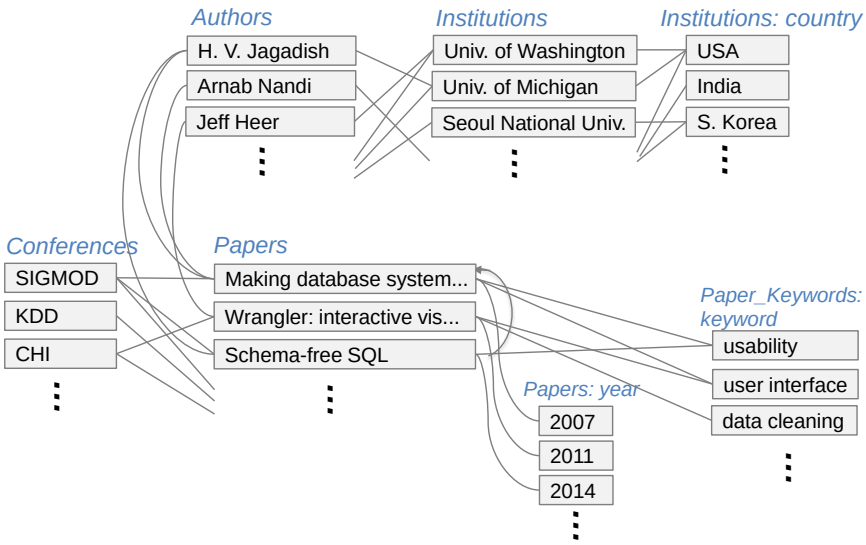


Figure 7.5: A part of the TGDB instance graph constructed from the academic dataset that follows the schema in Figure 7.4. Node types shown in blue italic font.

The typed graph model, similar to many graph data models [13, 70, 169], is much more effective for conveying a conceptual understanding of the mini-world represented in databases than the relational model. As it abstracts relational databases, users can ignore the logical and physical representation of data. Users can also easily understand the structure of data, since nodes always represent entities and edges represent relationships, Unlike TGM, the relational model is a mixture of entities, relationships, and multivalued attributes. Although some existing graph models are more expressive for representing a variety of relationships (e.g., hierarchical parent-child relationships among entities), we simply use nodes and edges to focus on making the semantics of the underlying relations more explicit by mapping to entities and relationships that they represent in the real world.

Relational databases can be translated into the TGDB schema and instance graphs in a near-automatic process. We adapt the reverse engineering literature pertaining to translating relational databases into several graph-style models [23, 42, 158]. Our procedure includes an analysis of a relational schema based on primary keys, foreign keys, and cardinalities for classifying tables into several categories, and a series of actions that create the

Table 7.1: Categories of node and edge types based on how they are translated from relational schema

Form	Source	Determining factor for mapping from a relational table
Node types	Entity tables	Relation with a single-attribute primary key
	Multi-valued attributes	Relation with two attributes; one of them is a foreign key of an entity relation
	Single-valued categorical attributes	Attribute of low cardinality
Edge types	One-to-many relationships	Foreign key between two entity relations
	Many-to-many relationships	Relation with a composite primary key; both are foreign keys of entity relations
	Multi-valued attributes	From an entity table to a multi-valued attr.
	Single-valued categorical attributes	From an entity table to a categorical attr.

schema graph. Table 7.1 summarizes the categories of node and edge types based on how they are determined from relational schema. Figures 7.4 and 7.5 illustrate a schema graph and a part of the instance graph constructed from an academic publication database whose schema is shown in Figure 7.3.

7.5 *ETable* Presentation Data Model

We present our *ETable* presentation data model for usable exploration of entities and relationships in databases.

7.5.1 Enriched Table

A query result in the *ETable* model is presented as an enriched table, which we also call *ETable*. An *ETable* R has a set of columns $\mathcal{A}$ and consists of a set of rows $r \in R$. The columns are categorized into two types: *single-attribute columns* and *entity-reference columns*. The value of the single-attribute column $r[A]$ is atomic as it is in the relational

model. The value of the entity-reference column $r[A]$ contains a single or a set of *entity references*. The entity reference refers to another node in the database instance graph. Unlike a foreign key in the relational model, each entity reference is shown as a clickable label, similar to a *hyperlink* on a webpage. Just like how a hyperlink's hypertext describes the webpage that the link points to (instead of its URL), for example, *ETable* represents an author's entity reference by the author name (instead of the author ID).

The entity-reference columns present rich information spread across multiple relations within a single enriched table. While a foreign key attribute in the relational model contains only a single reference for a many-to-one relationship because of the first normal form, an entity-reference column can represent one-to-many relationships, many-to-many relationships, or multivalued attributes in a single column. Furthermore, the entity-reference column has advantages over the nested relational model which requires much screen space as it squeezes another table into cells, leading to inefficient browsing. Unlike the nested model, *ETable* presents clickable labels that compactly show information and allow users to further explore relevant information.

7.5.2 *ETable* Specification

An *ETable* can be specified by selecting specific elements of the TGDB database schema and instance graphs introduced in the previous section.

Definition 3. *ETable* Query Specification. An *ETable* R is specified by a *query pattern* Q, which is a tuple $(\tau_a, T, P, \mathcal{C})$.

1. **Primary node type** τ_a: It is one of the node types in the schema graph. Each row of *ETable* will represent a single node instance of the primary node type.

2. **Participating node types** T: It is a set of node types chosen from the node types in the schema graph (i.e., $T = \{t_1, ..., t_{n_T}\}, \forall t_i \in \mathcal{T}$). It must contain the primary node type τ_a (i.e., $\tau_a \in T$). It determines the scope of data instances and is similar to a

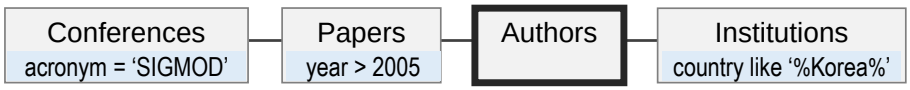


Figure 7.6: An example query pattern in a diagrammatic notation. It represents a query that finds a list of researchers who have published papers at SIGMOD after 2005 and are currently working at institutions in Korea.

set of relations in SQL FROM clauses. A node type in the schema graph can exist multiple times in the participating node types, like a relational algebra expression can contain the same relation multiple times.

3. **Participating edge types** P: It is a set of edge types selected from the schema graph (i.e., $P = \{p_1, ..., p_{n_P}\}, \forall p_i \in \mathcal{P}$). It connects the participating nodes types, thus all the source and target nodes of these edges should exist in the participating node types (i.e., $source(p_i) \in T \wedge target(p_i) \in T, \forall p_i \in \mathcal{P}$).

4. **Selection conditions for node types** $\mathcal{C}$: It is a set of selection conditions $\mathcal{C} = (C_1, ..., C_{n_T})$ applied to each of the participating node types, i.e., C_i applies to $t_i \in T$.

A query pattern can be represented as an acyclic graph where one of the nodes is marked as a primary node type and any node can have selection conditions. For example, the query pattern in Figure 7.6 represents a query that produces a list of researchers who have published papers at SIGMOD after 2005 and are currently working at institutions in Korea.

7.5.3 Incremental Query Building with Primitive Operators

In *ETable*, a query pattern is constructed and updated by *primitive operators*. Each operator builds on an existing *ETable* query to generate a new, updated *ETable* query. This subsection describes these operators in detail. Then Section 7.6.1 will describe how users' actions performed on the *ETable* user interface will invoke these operators. Formally, given

an *ETable* specification $Q(\tau_a, T, P, \mathcal{C})$, each of the following operator creates a new specification $Q'(\tau'_a, T', P', \mathcal{C}')$, except the *Initiate* operator which creates a new *ETable* from scratch.

1. **Initiation.** A new *ETable* can be created by selecting one of the node types τ_k in the schema graph. Its result lists the corresponding nodes.

$$Initiate(\tau_k) = Q'$$
$$\text{where } \tau'_a = \tau_k,\ T' = \{\tau_k\},\ P' = \{\},\ \text{and } \mathcal{C} = \{\}.$$

2. **Selection.** *ETable* rows can be filtered based on their columns, similar to the selection operator in the relational model. Applying a selection condition C_k to the primary node type τ_a filters the rows of the current *ETable*.

$$Select(C_k, Q) = Q'$$
$$\text{where } \tau'_a = \tau_a,\ T' = T,\ P' = P,\ \text{and } C'_a = C_k.$$

3. **Adding a node type.** Another node type can be added to a query pattern to examine how it is related to the current primary node type. It corresponds to adding a join operator in the relational model. Selecting one of the node types that are linked to the primary node type τ_a by an edge type ρ_k (i.e., $source(\rho_k) = \tau_a$), adds it to the participating node types in the current query Q.

$$Add(\rho_k, Q) = Q'$$
$$\text{where } \tau'_a = target(\rho_k),\ T' = T \cup \{target(\rho_k)\},$$
$$P' = P \cup \{\rho_k\},\ \text{and } \mathcal{C}' = \mathcal{C} \cup \{\}.$$

4. **Shifting focus to another participating node type.** The primary node type τ_a can be changed to one of the other participating node types τ_k. It can be thought of as

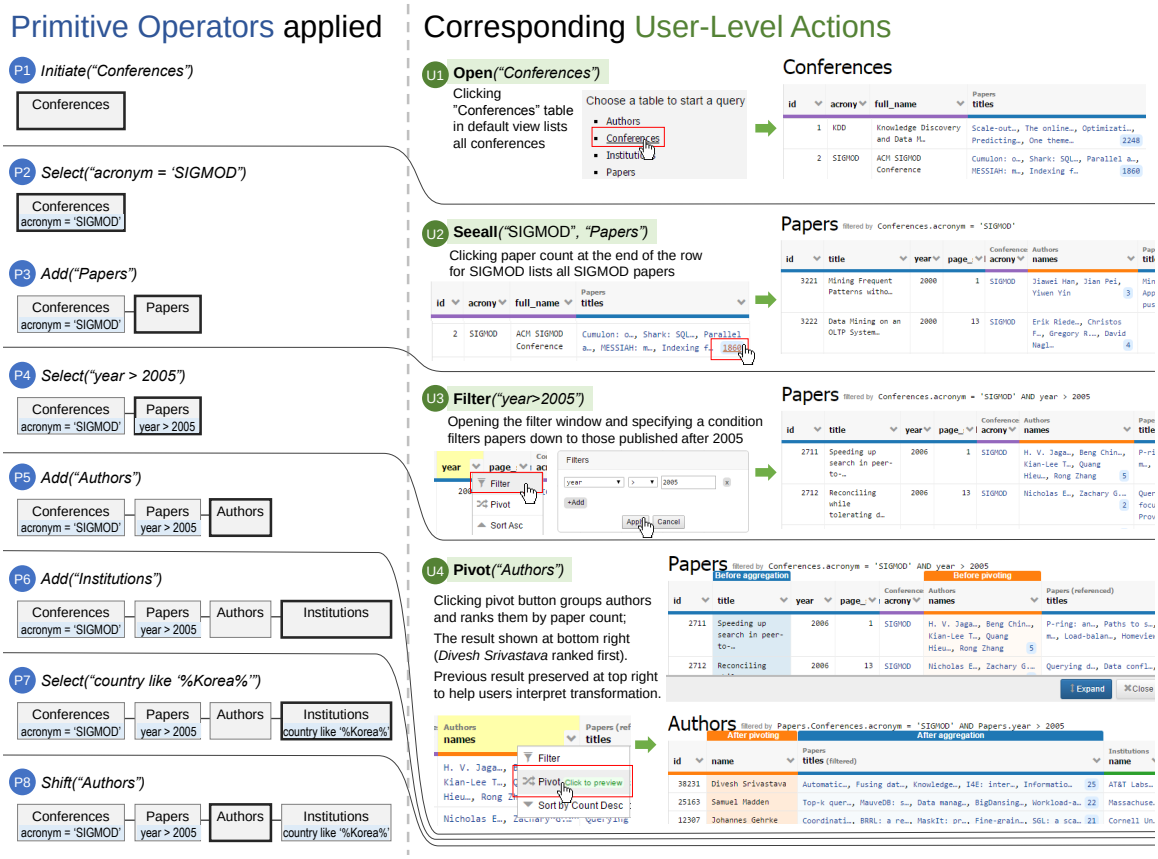


Figure 7.7: An example of incrementally building a complex query: *find a list of researchers who have published papers at SIGMOD after 2005 and are currently working at institutions in Korea*. **Left**: constructing the query through a series of *ETable* primitive operators. **Right**: corresponding user actions in the interface that invoke the operators (Section 7.6.1 describes the user-level actions in detail). User actions for the operators P6-P8, similar to the others shown in the figure, are omitted for brevity.

representing the current join result from a different angle.

$$Shift(\tau_k, Q) = Q'$$

where $\tau'_a = \tau_k$, $T' = T$, $P' = P$, and $\mathcal{C}' = \mathcal{C}$.

The above primitive operators enable us to build any complex queries by incrementally specifying the operators one-by-one. Figure 7.7 (left) illustrates the query construction process consisting of 8 operators. A new query pattern can be created with *Initiate*; Selection

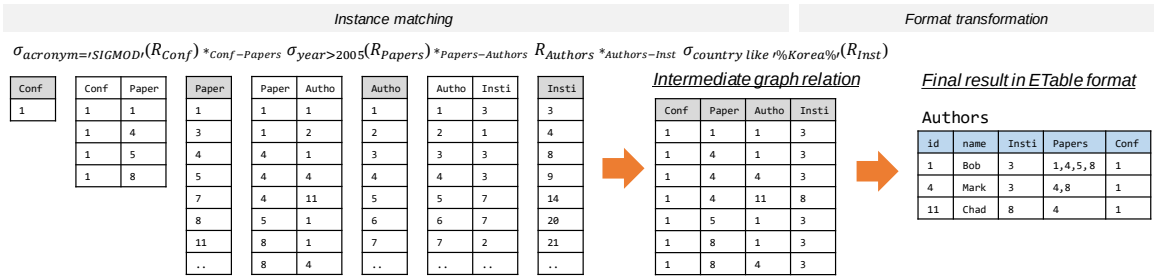


Figure 7.8: *ETable* query execution process consists of two steps: (1) the instance matching step which extracts matched instances from the instance graph and (2) the format transformation step which transforms the instances into the *ETable* format.

conditions can be added with *Select*, just like writing expressions in WHERE clauses in SQL; and node types can be added with *Add*, just like adding relations to FROM clauses and setting one of them as a GROUP BY attribute. Also, the primary node type can be changed with *Shift*, similar to changing the GROUP BY attribute. A sequence of these operators specified constitutes a query pattern in the *ETable* model. These operators can be invoked by users on the user interface with *user-level actions*, which we will describe details in Section 7.6.1. The right side of Figure 7.7 shows how users can specify the same query through the user interface.

7.5.4 Query Execution

A query pattern is executed to produce a result in the *ETable* format. The execution process is divided into two steps: *instance matching* and *format transformation*. The first step extracts matched node instances from the TGDB instance graph, and the second step transforms a result from the first step into the *ETable* format.

Instance Matching

The instance matching process finds a set of matched instances for a given query pattern. Formally, it returns a *graph relation* R^G, which consists of a set of tuples, each of which

contains a list of node instances in the database instance graph. The graph relation is generated with an *instance matching function* $m(Q)$, which consists of a series of operations. The operations constitute primitives which make up a *graph relation algebra*.

A *graph relation* R^G, similar to a relation in the relational model, consists of a set of tuples with a set of attributes. The schema of the graph relation is defined as a set of node types $\mathcal{A} = (A_1, ..., A_n)$ where $A_i \in \mathcal{T}$. In other words, each attribute A_i corresponds to a node type. The node type τ_j determines the domain of the attribute (i.e., $domain_i = \{v | v \in V_j\}$). A *base graph relation* is defined as a graph relation with a single attribute. In other words, each node type $\tau_1, ..., \tau_n$ produces a base graph relation $R_1^G, ..., R_n^G$. A *non-base graph relation* can be created by applying the following graph relation operators to the base graph relations.

1. **Selection.** It filters tuples of a graph relation R using a selection condition C_i applicable to one of the node types A_i.

$$\sigma_{C_i}(R^G) = \{r | r \in R^G \wedge r[A_i] \text{ satisfies } C_i\}.$$

2. **Join.** It joins two graph relations R_1 and R_2 using edge types ρ_k. The attributes of the created graph relation is a concatenation of the attributes of the two graph relations.

$$R_1^G *_{\rho_k} R_2^G = \{(r_1, r_2) | r_1 \in R_1^G \wedge r_2 \in R_2^G \\ \wedge source(\rho_k) \in \mathcal{A}_1 \wedge target(\rho_k) \in \mathcal{A}_2\}.$$

   We use a symbol, $*$, to differentiate it from the relational correspondence, $\bowtie$, and not to be confused with natural join.

3. **Projection.** It removes all attributes of the graph relations except the given attribute. Duplicated rows are eliminated.

$$\Pi_{A_i}(R^G) = \{r[A_i] | r \in R^G\}.$$

These operators enable us to define an instance matching function $m(Q)$. In fact, this

function only requires the *Selection* and *Join* operators: the *Projection* operator will be used later in the format transformation step.

Definition 4. Instance Matching. Given a *ETable* query pattern $Q(\tau_a, T, P, \mathcal{C})$, a matching function m returns a graph relation R^G containing node instances in the instance graph $\mathcal{G}_I$.

$$m(Q) = \sigma_{C_1}(R_1^G) *_{p_1} \sigma_{C_2}(R_2^G) *_{p_2} ... *_{p_{n-1}} \sigma_{C_n}(R_n^G),$$

where R_i^G is a base graph relation obtained from a node type $t_i \in T$, i.e., $R_i^G = \{v | v \in V \wedge type(v) = t_i\}$, $C_i \in \mathcal{C}$ is a selection condition for R_i, and $p_i \in P$ is one of the edge types that joins graph relations on both sides, i.e., $p_i = \{p | p \in P \wedge source(p) \in \{t_1, ...t_i\} \wedge target(p) \in \{t_{i+1}, ...t_n\}\}$.

Figure 7.8 (left) illustrates the instance matching process. It returns a graph relation, which is an intermediate format to be transformed into the *ETable* format.

Format Transformation

A graph relation obtained from the *instance matching* function is transformed into the *ETable* format. We describe how rows and columns of *ETable* are determined from it.

The rows of *ETable* consist of nodes of the primary node type, filtered by all selection conditions in the query pattern. They are extracted from the instance matching result:

$$R = \{v | v \in \Pi_{\tau_a}(m(Q(\tau_a, T, P, \mathcal{C})))\}.$$

Given the result of the instance matching function, all attributes except the attribute representing the primary node type are discarded, and then, each of distinct node in that column becomes a row.

ETable has three types of columns to present rich information for each row. In addition to the attributes of the primary node types, which we call *base attributes* $\mathcal{A}_b$, we introduce two other types of columns for presenting a set of entity references: *participating node columns*, $\mathcal{A}_t$, and *neighbor node columns*, $\mathcal{A}_h$.

1. **List of base attributes** $\mathcal{A}_b$: It is a full set of the attributes A of the primary node type τ_a. The value of the column $A_j \in \mathcal{A}_b$ would be a single value:

$$r[A_j] = v[A_j].$$

2. **List of participating node types** $\mathcal{A}_t$: It is a set of all the node types T in the query pattern, except the primary node type τ_a, i.e., $\mathcal{A}_t = \{\tau | \tau \in T \wedge \tau \neq \tau_a\}$. The value of the column $A_j \in \mathcal{A}_t$ would be a set of entity references:

$$\begin{aligned} r[A_j] = &\{u | u \in V \wedge A_j = type(u) \\ &\wedge \Pi_{type(u)} \sigma_{\tau_a = r}(m(Q))\}. \end{aligned}$$

3. **List of neighbor node types** $\mathcal{A}_h$: It is a set of all the neighboring node types of the primary node type τ_a in the schema graph regardless of the query pattern, i.e., $\mathcal{A}_h = \{(\rho, \tau) | \tau \in \mathcal{T} \wedge \rho \in \mathcal{P} \wedge source(\rho) = \tau_a \wedge target(\rho) = \tau\}$. The value of the column $A_j \in \mathcal{A}_h$ would be a set of nodes references:

$$\begin{aligned} r[A_j] = &\{u | u \in V \wedge e \in E \wedge A_j = (type(e), type(u)) \\ &\wedge u = target(e) \wedge r = source(e)\}. \end{aligned}$$

Figure 7.8 (right) illustrates the results produced from the format transformation process. The first two columns are base attributes, and the rest of the columns are participating node columns. We omit neighbor node columns as some of these columns are the same as the participating node columns.

By transforming the graph relation into the *ETable* format, we compactly present join query results without duplications. Each row of *ETable* is uniquely determined by a node of a primary node type. The participating node columns show all the other entity types in the query pattern with respect to the primary node type. This transformation process is similar to setting one of the relations as a GROUP BY attribute in SQL, but while GROUP BY aggregates the corresponding instances into a single value (i.e., COUNT, AVG), *ETable*

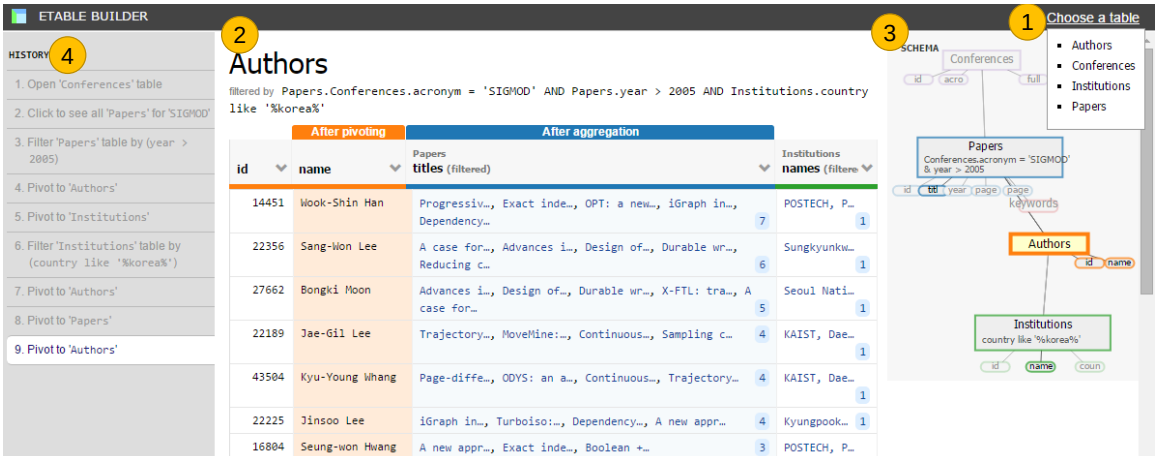


Figure 7.9: The *ETable* interface consists of (1) the default table list for initiating a query, (2) the main view presenting query results, (3) the schema view showing a query pattern, and (4) the history view listing operators specified by users. Users can build queries and explore databases by directly interacting with the interface.

presents a list of the corresponding instances as entity references. The neighbor node columns are also useful for describing the rows of the *ETable*, although information in these columns is not obtained from the graph relation. These columns enable users to browse one-to-many or many-to-many relationships. Moreover, they provide users with a preview of possible new join operations as it presents all the join candidates. For instance, a *ETable* in Figure 7.1 consists of many neighbor node columns (e.g., Authors) that helps users browse rich information about each paper.

7.6 Interface and System Design

ETable's interface (Figure 7.9) consists of four components: (1) the default table list, (2) the main view, (3) the schema view, and (4) the history view. The *default table list* presents a list of entity types in the schema graph. Users can pick one from the list to initiate a query. The *main view* presents an *ETable* executed based on a query pattern which is graphically shown over the *schema view*. Users can directly interact with the main view to update the

current query. The list of actions specified by users is presented on the *history view*, which allows users to revert to a previous state.

7.6.1 User-Level Actions

Users can update the current query pattern by directly interacting with *ETable* via **user-level actions**. As shown in Figure 7.7, these actions in turn invoke their corresponding primitive operators (discussed in Section 7.5.3).

1. **Open a new table.** Users can open a new table by clicking a node type τ_k on the default table list. The action invokes the $Initiate(\tau_k)$ operator (Fig 7.7: action U1).

$$\textbf{Open}(\tau_k) = Initiate(\tau_k).$$

2. **Filter.** Users can filter the rows of the current *ETable* by inducing selection conditions via a popup window at the column header (Fig 7.7: action U3). Besides the base attributes, users can also filter rows by the labels of the neighbor nodes columns (e.g., authors' names), which is translated into subqueries. We currently provide only a conjunction of predicates, but it is straightforward to provide disjunctions and more operations. The action invokes the *Select* operator.

$$\textbf{Filter}(C, R) = Select(C, R).$$

3. **Pivot.** Users can change the primary node type by clicking the pivot button on the context menu for neighbor or participating node columns. It calls the *Add* operator if the column is the neighbor node type (Fig 7.7: action U4); it performs the *Shift* operator if it is the participating node type.

$$\textbf{Pivot}(\rho_l, R) = Add(\rho_l, R),$$
$$\text{or } \textbf{Pivot}(\tau_k, R) = Shift(\tau_k, R).$$

4. **See a particular node.** When users are interested in one of the entity references, they

can click it to create a new *ETable* consisting of a single row presenting the clicked entity. Unlike the above actions, it invokes two primitive operators: it initiates a new *ETable*, and then perform the *Select* operator to show the single node. For the clicked node v_k:

$$\textbf{Single}(v_k, R) = Select(C, type(v_k), Initiate(type(v_k)),$$
$$\text{where } C = \{u | u = v_k\}.$$

5. **See all related nodes.** When users are interested in a full list of entity references, they can click a number (i.e., entity reference count) in the right corner of a cell (Fig 7.7: action U2). It also encapsulates two primitive operators. The operators invoked are different depending on whether the selected column is *neighbor* or *participating* node column. For the *neighboring node column* ρ_l of v_k:

$$\textbf{Seeall}_h(v_k, \rho_l, R) = Add(\rho_l, Select(C, type(v_k), R)),$$
$$\text{where } C = \{u | u = v_k\},$$

and for the *participating node column* t_l:

$$\textbf{Seeall}_t(v_k, t_l, R) = Shift(t_l, Select(C, type(v_k), R)), R)),$$
$$\text{where } C = \{u | u = v_k\}\}.$$

ETable supports additional actions that help with database exploration, such as: (1) **Sort rows** based on the values in a column; (2) **Hide/show columns** to reduce visual complexity in the interface; and (3) **Revert to previous queries** via the left history panel.

7.6.2 Architecture

ETable system uses a three-tier architecture, consisting of (1) an interactive user interface front-end that can run in any modern web browsers, written in HTML, JavaScript, and

D3.js[2]; (2) a Python-based application server; and (3) a PostgreSQL database backend. The PostgreSQL database stores TGDB schema and instance graphs in four relational tables: `nodes`, `edges`, `node_types`, and `edge_types`. A query pattern for *ETable* is translated into SQL queries that operate on the PostgreSQL database. To efficiently perform queries, we partition a long SQL query into multiple queries consisting of a fewer number of relations to be joined (i.e., each for a single entity-reference column) and merge them.

7.7 Evaluation: User Study

To evaluate the usability of *ETable*, we conducted a user study that tests whether users can construct queries quickly and accurately. We compared *ETable* with Navicat Query Builder.[3] Navicat is one of the most popular commercial database administration tools with a graphical query building feature. Graphical builders such as Navicat Query Builder have been commonly used as baseline systems in database usability research [116, 131, 19].

7.7.1 Experimental Design

Participants. We recruited 12 participants from our university through advertisements posted to mailing lists at our institution. All were graduate students who had taken at least one database course or had industry experience using database systems. The participants rated their experience in SQL, averaging at a score of 4.67 using a 7-point Likert scale (ranged from 3 to 6) with 1 being "having no knowledge" and 7 being "expert", which means most participants considered themselves non-expert database users. None of them had used the graphical query builder before. Each participant was compensated with a $15

[2]D3.js, `https://d3js.org/`
[3]`http://www.navicat.com/`

gift card.

Dataset. We used an academic publication dataset used throughout this paper, which we collected from DBLP[4] and ACM Digital Library.[5] It contains about 38,000 papers from 19 top conferences in the areas of databases (e.g., SIGMOD), data mining (e.g,. KDD), and human-computer interaction (e.g., CHI), since 2000. A relational schema was designed using standard design principles, resulting in 7 relations with 7 foreign keys as depicted in Figure 7.3. As the main focus of this evaluation is on *ETable*'s usability, this dataset creates a sufficiently large and complex database for such purpose.

Procedure. Our study followed a *within-subjects design* with two conditions: the *ETable* and Navicat conditions. Every participant first completed six tasks in one condition and then completed another six tasks in the remaining condition. The orders of the conditions were counterbalanced, resulting in 6 participants in each ordering. We generated two matched sets of tasks (6 tasks in each set) differing only in their specific values used for parameters such as the title of the paper. Before the participants were given the tasks to carry out for each condition, they went through a 10-minute tutorial for the tool they would use. For each task, the participants could ask clarifying questions before starting, and they had a maximum of 5 minutes to complete each task. After the study, they completed a questionnaire for subjective ratings and qualitative feedback. Each study lasted for about 70 minutes. Participants completed the study using Chrome browser, running on a Windows desktop machine, with a 24-inch monitor at a 1920x1200 resolution.

Tasks. We carefully generated two matched sets of 6 tasks that cover many database exploration and querying tasks. Table 7.2 shows one set (the other set is similar). The tasks fall into three categories: finding attribute values (Tasks 1 & 2); filtering (Tasks 3 & 4); aggregation (Tasks 5 & 6). The tasks were designed based on prior research studies and their categorization of tasks. Specifically, our categories are based on those used in

[4]DBLP, `http://dblp.uni-trier.de/`
[5]ACM Digital Library, `http://dl.acm.org/`

Table 7.2: List of tasks. Task 1 & 2 retrieve attribute values, task 3 & 4 filter entities, task 5 & 6 perform aggregations.

Task	Category	#Relations
1. Find the year that the paper titled 'Making database systems usable' was published in.	Attribute	1
2. Find all the keywords of the paper titled 'Collaborative filtering with temporal dynamics'.	Attribute	2
3. Find all the papers that were written by 'Samuel Madden' and published in 2013 or after.	Filter	3
4. Find all the papers written by researchers at 'Carnegie Mellon University' and published at the KDD conference.	Filter	5
5. Which institution in South Korea has the largest number of researchers?	Aggregate	2
6. Find the top 3 researchers who have published the most papers in the SIGMOD conference.	Aggregate	4

database and HCI research [7, 110], and our tasks vary in difficulty as in [109].

Measurements. We measured participants' task completion times. If a participant failed to complete a task within 5 minutes, the experimenter stopped the participant and recorded 300 seconds as the task completion time. After completing tasks for both conditions, the participants filled out a post-questionnaire that asked for their subjective ratings about *ETable* (10 questions) and their subjective preference between two conditions (7 questions).

7.7.2 Results

Task completion times. The average task times for *ETable* were faster than those for Navicat for all six tasks. Figure 7.10 summarizes the task time results. We performed two-tailed paired t-tests. The differences were statistically significant for Tasks 1, 3, 5, and 6 ($p < 0.005$) and marginally significant for Tasks 2 and 4 ($p = 0.052$, $p = 0.053$, respectively). The results of Task 2 may be explained by an outlier participant who did not understand the requirement that each row of the final results must represent a different

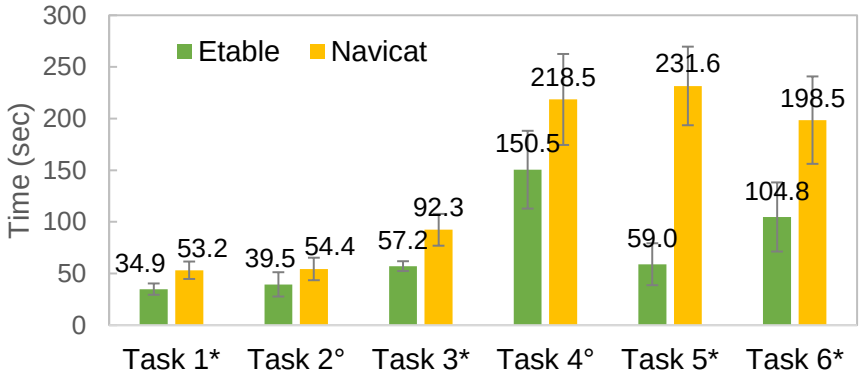


Figure 7.10: Average task completion time for each task. The error bars represent 95% confidence intervals for the mean. Participants performed the tasks faster with *ETable* than with Navicat. The * and ° symbols indicate 99% and 90% statistical significance in the two-tailed paired t-tests, respectively.

keyword. Although Task 4 involves the highest number of operations that require participants to spend significant time in interpreting intermediate results before applying the next operators, *ETable* helped participants complete this task over 30% faster than Navicat.

The task completion times for *ETable* generally have low variance. The larger variance in Navicat is mainly due to syntax errors that the participants faced. Many participants, who are non-database experts, could not recall some SQL syntax and had trouble debugging errors. In particular, they had trouble specifying GROUP BY queries in Navicat. For example, many participants did not specify a GROUP BY attribute in their SELECT clauses in their first attempts. We also observed that many Navicat participants were overwhelmed by the complexity of the syntax of join queries [81] and preferred to specify new SQL queries from scratch instead of debugging existing ones when their original queries failed. Unlike graphical query builders such as Navicat, *ETable* helps nonexperts gradually build complex queries without having to know the exact query syntax.

Subjective ratings. We asked participants to rate various aspects of *ETable* using 7-point Likert scales (7 being "strongly agreed"). Their subjective ratings were generally very positive (see Table 7.3). In particular, all participants found *ETable* easy to learn (i.e.,

Table 7.3: Subjective ratings about *ETable* using 7-point Likert scales (7: *Strongly Agreed*. 1: *Strongly Disagreed*).

Question	**Avg.**
1. Easy to learn	6.42
2. Easy to use	6.33
3. Helpful to locate and find specific data	6.25
4. Helpful to browse data stored in databases	6.67
5. Helpful to interpret and understand results	5.58
6. Helpful to know what type of information exists	6.00
7. Helpful to perform complex tasks	6.00
8. Felt confident when using *ETable*	5.92
9. Enjoyed using *ETable*	6.42
10. Would like to use software like *ETable* in the future	6.50

rated 6 or 7), and almost all participants (11/12) found *ETable* easy to use and helpful for browsing data in databases. They also enjoyed using *ETable* (10/12) and would like to use software like *ETable* in the future (11/12). In response to the "*helpful to interpret and understand results*" question, one participant commented that "*there are too many attributes ..., which is not easy to interpret.*" To address this, as future work, we plan to develop techniques to rank and select the most important columns to show whenever a table has a large number of columns [188].

We also asked participants to compare *ETable* and Navicat in 7 aspects. All participants indicated that *ETable* was easier to learn and was more helpful in browsing and exploring data. A majority of participants liked *ETable* more (11/12) and found it easier to use (10/12). They would choose to use *ETable* in the future (10/12) and felt more confident using it (8/12). Half of the participants answered that *ETable* is more helpful in finding specific data than Navicat. This result was expected because *ETable*'s innovation focuses more on supporting data exploration.

Qualitative feedback. We asked participants about the features they liked about *ETable*. Many participants (9/12) explicitly mentioned the "pivot" feature. They said that the pivot feature enabled them to easily specify complex join queries. One participant said "*I also*

loved the pivot feature ... having multiple pivots throughout the course of forming a query. I messed up a query, but could still find the right answer by doing an appropriate pivot." In addition, many participants said that *ETable* provides an intuitive view to users. One said "*It is easy to see data from the perspective of what the users want to see/retrieve ...*" Another said "*Visually, I was able to see ... the effects of the SQL operations, which made it easier to use and verify intermediate results.*"

7.8 Expressiveness

This section discusses the expressiveness of the *ETable* model. We will first express the overall functionality of the *ETable* queries as a general SQL query pattern. By doing so, we will show how typical join queries can be translated into *ETable* queries, through multiple steps (similar to [116, 35]), demonstrating *ETable*'s expressiveness. Any join queries involving only FK-PK relationships on a relational database schema that meets *ETable*'s assumptions. can be translated into an *ETable* query that operates on TGDB.

The overall functionality of *ETable* queries can be expressed as the following general SQL query pattern:

```
SELECT τ_a.*, ent-list(t_1), ent-list(t_2), ...
FROM t_1, t_2, ...
WHERE source(p_1) = target(p_1) AND source(p_2) =
      target(p_2) AND ... AND C_1 AND C_2 AND ...
GROUP BY τ_a;
```

where `ent-list` presents a list of corresponding entity references, similar to the `json_agg` operator in PostgreSQL.[6] Each of the four components in an *ETable* query (i.e., primary node type τ_a, node types T, edge types P, and selection conditions $\mathcal{C}$) maps to a clause in

[6] `http://www.postgresql.org/docs/9.4/static/functions-aggregate.html`

SQL: primary node type to GROUP BY clause; node_types to FROM clause; edge_types to join conditions; selection conditions to WHERE clause.

Following the above mappings, we now follow the approach similar to that in [116, 35] to show that *ETable* can expressively handle typical join SQL queries, through a step-by-step translation. That is, for any SQL join query following the above pattern, there exists an equivalent *ETable* query.

1. Transforms a relational algebra join expression ($R \bowtie R \bowtie ...$) to a graph relation correspondence $R^G * R^G * ...$ (described in Section 5.4) by analyzing the list of relations in the FROM clause, and the join conditions in the WHERE clause. (Each R^G is a node type; each $*$ an edge type.)

2. Applies the original selection conditions to the TGDB's node types;

3. If there is a *group by* attribute, transform it to the graph's primary node type; otherwise, if no *group by* attribute exists, arbitrarily set a primary node type.

ETable can express typical join queries consisting of the core relational algebra (i.e., relational algebra expression that does not contain set operations), which accounts for a large number of the database workloads. *ETable* additionally lets users choose a *primary node type* from the list of selected relations, and introduces the *entity-reference columns* (i.e., represented as `ent-list` in the above SQL pattern) to effectively present join queries. This paper focuses on the critical usability challenge that arises when joining several tables. In our future work, we plan to further increase *ETable*'s expressiveness of the presentation model to the full set of operators in the relational algebra, through introducing additional operators to support more complex queries (e.g., set operations, complex aggregations).

7.9 Conclusions

We proposed *ETable*, a new presentation data model for interactively exploring relational databases. The enriched table representation of *ETable* generates a holistic, interactive view of databases that helps users browse relevant information at an entity-relationship level. By directly interacting with the interface, users can iteratively specify operators, enabling them to incrementally build complex queries and navigate databases. *ETable* outperformed a commercial graphical query builder in a user study, in both speed and subjective ratings across a range of database querying tasks.

This work takes a first step towards developing a practically usable, interactive interface for relational databases, and opens up many interesting opportunities. Future research directions include: (1) incorporating more operations to further improve expressive power (e.g., set operations); (2) accelerating the execution speed of updated queries (e.g., by reusing intermediate results); (3) leveraging machine learning techniques to rank and select important columns to display. The above ideas could usher a new generation of interactive database exploration tools that will benefit all database users.

CHAPTER 8

CONCLUSIONS

In summary, my dissertation addresses the fundamental and practical challenges in the understanding of machine learning models by developing scalable, interactive visual analytics tools that help users explore and interact with models through data. My work contributes to novel visualization tools, new data analytics paradigms, user interaction workflows, and scalable and accessible approaches. I believe my research advances human understanding of artificial intelligence, accelerate their development, and increase people's trust in this new technology.

8.1 Contributions

My thesis makes research contributions through multiple major fronts.

- **New design principles**: My dissertation contributes novel design principles for developing interactive visualization tools for complex deep learning models. Our tools provide users with *both a **high-level visual overview** of the models and **interactive methods to drill down into details** of the models or datasets*. In *ActiVis*, users can start their exploration with a graph-structured overview, and then dive into details of neurons' activation (Chapter 3). Also, *GAN Lab*'s coordinated views help users perform experimentations while visualizing a model's architecture (Chapter 6).

- **Novel data exploration models**: We contribute new ways to analyze how datasets affect machine learning results. The *MLCube* framework enables users to flexibly

specify *data subset* by considering every part of a machine learning pipeline and interactively drill down into specific subsets for in-depth exploration (Chapter 4). The *ActiVis* system further ***unifies the subset-level analysis with the instance-level analysis***, which allows to ***scale*** to large-scale datasets (Chapter 3). *ETable* also contributes new models for exploring data (Chapter 7).

- **New scalable, deployed systems**: We present new scalable systems for interpreting large-scale machine learning systems. *ActiVis*'s multiple scalable techniques enabled it to ***scale to industry-scale datasets and models*** and ***deploy*** to Facebook's internal machine learning platform (Chapter 3). *MLCube*'s scalable system design also led to a deployment by Facebook and influenced Google's open-source library (Chapter 4).

- **New broadly accessible approaches**: Our browser-based visualization tools significantly broaden public's access to modern AI technologies. *GAN Lab* overcomes a major practical challenge in deploying interactive tools for deep learning, by enabling users to *learn about models by playfully training and* ***experimenting*** *with them on web browser* (Chapter 6). *FairVis* also allows users to audit fairness of machine learning models on their browser (Chapter 5). Both tools have also been open-sourced.

8.2 Future Research Directions

My long-standing research goal is to bring human-centered approaches to the field of AI and data science. I have taken the first important steps toward this goal with my thesis research. For the road ahead, I hope to broaden and deepen this investigation. I plan to pursue this in the following thrusts.

Model debugging with visual guidance. While visualization tools promote people's understanding of machine learning models, they often do not directly support the task of building well-performing models. This important task for practitioners is difficult even for

experts, due to the countless combinations of parameters that need tuning. They often have to iteratively try different heuristics-based strategies to improve models, which is tedious and error-prone. I believe new types of visualization tools can help them refine and debug their models by guiding them to identify problems and discover actionable insights for debugging. These tools will combine computational and interactive methods: scalable data mining techniques automate the discovery of problematic cases; interactive tools guide users to explore debugging strategies and make decisions.

Interactive model building for everyone. A growing number of non-experts who do not write code not only want to learn artificial intelligence (AI), but also want to build machine learning models for their products and data. Large technology companies have started providing these users with web-based services for building machine learning models without any coding. They include automated approaches to building machine learning models called *AutoML* [111] and graphical interfaces for creating machine learning models. However, it is very challenging to develop interfaces for such systems that are easy for non-experts to learn and to use. I envision the next generation of interactive systems for these users to easily interact with machine learning models that use large datasets with the help of visual explanations to avoid the users to use AI as black-boxes. I believe this new accessible way of building machine learning models will broaden people's access to AI technologies and ensure their appropriate use.

New interaction paradigms between human and AI systems. The advent of deep learning models is rapidly reshaping many existing data-driven systems that have a long history of research and development, such as database systems and search engines. For example, traditionally, web search engines take only short text queries, however, they now recognize long natural language queries, images, voice, and more. This has led us to re-think their interaction methods that involve both computational and usability challenges. For instance, when a system fails to recognize a user's voice query, we may want to design interfaces that actively solicit user feedback to iteratively refine the query. Therefore,

many opportunities exist in studying new interaction paradigms between these data-driven intelligent systems and human, and this may require collaboration across multiple areas.

Ensuring AI working for our society. As AI-powered systems continue to make important decisions across social domains, it is becoming more important to ensure AI works for everyone and our society. For example, as we discussed in this dissertation, many researchers have recognized that machine learning models can be unfair and have developed new methods that reveal unfairness and mitigate problems. Besides fairness, there are a wide range of aspects, such as accountability, transparency, and safety. I hope our research community works together with people from different backgrounds, to identify potential problems, study AI's influences, and ensure it works for people and society.

8.3 Concluding Remarks

My dissertation pushes the frontier of AI through human-centered approaches, contributing novel paradigms, methods, and tools that advance people's understanding of AI, accelerate its development, and increase their access to new technologies. With my experience and knowledge across interactive visualization, data science, and machine learning, I hope to accelerate innovation across these disciplines, making positive impacts to people's everyday lives and our society.

REFERENCES

[1] D. Abadi, R. Agrawal, A. Ailamaki, M. Balazinska, P. A. Bernstein, M. J. Carey, S. Chaudhuri, J. Dean, A. Doan, M. J. Franklin, *et al.*, "The beckman report on database research," *ACM SIGMOD Record*, vol. 43, no. 3, pp. 61–70, 2014.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[3] A. Abdulkader, A. Lakshmiratan, and J. Zhang, "Introducing DeepText: Facebook's text understanding engine," 2016. Accessed: June 26, 2017. [Online]. Available: `https://code.facebook.com/posts/181565595577955/introducing-deeptext-facebook-s-text-understanding-engine/`.

[4] S. Abiteboul, R. Agrawal, P. Bernstein, M. Carey, S. Ceri, B. Croft, D. DeWitt, M. Franklin, H. G. Molina, D. Gawlick, *et al.*, "The lowell database research self-assessment," *Communications of the ACM*, vol. 48, no. 5, pp. 111–118, 2005.

[5] A. Abouzied, J. Hellerstein, and A. Silberschatz, "DataPlay: Interactive tweaking and example-driven correction of graphical database queries," in *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST)*, ACM, 2012, pp. 207–218.

[6] S. Agrawal, S. Chaudhuri, and G. Das, "Dbxplorer: A system for keyword-based search over relational databases," in *Proceedings of the 18th International Conference on Data Engineering*, IEEE, 2002, pp. 5–16.

[7] R. Amar, J. Eagan, and J. Stasko, "Low-level components of analytic activity in information visualization," in *IEEE Symposium on Information Visualization (INFOVIS)*, IEEE, 2005, pp. 111–117.

[8] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza, "Power to the people: The role of humans in interactive machine learning," *AI Magazine*, vol. 35, no. 4, pp. 105–120, 2014.

[9] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh, "ModelTracker: Redesigning performance analysis tools for machine learning," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI)*, ACM, 2015, pp. 337–346.

[10] M. R. Anderson, D. Antenucci, V. Bittorf, M. Burgess, M. J. Cafarella, A. Kumar, F. Niu, Y. Park, C. Ré, and C. Zhang, "Brainwash: A data system for feature engineering.," in *6th Biennial Conference on Innovative Data Systems Research (CIDR)*, 2013.

[11] M. R. Anderson and M. Cafarella, "Input selection for fast feature engineering," in *Proceedings of the 32nd International Conference on Data Engineering (ICDE)*, 2016, pp. 577–588.

[12] P. Andrews, A. Kalro, H. Mehanna, and A. Sidorov, "Productionizing machine learning pipelines at scale," in *ML Systems Workshop at the 33rd International Conference on Machine Learning (ICML)*, 2016.

[13] R. Angles and C. Gutierrez, "Survey of graph database models," *ACM Computing Surveys*, vol. 40, no. 1, p. 1, 2008.

[14] J. Angwin, J. Larson, L. Kirchner, and S. Mattu, "Machine bias," *ProPublica*, 2016. Accessed: Mar. 31, 2019. [Online]. Available: `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing`.

[15] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," in *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, ACM, 1999, pp. 49–60.

[16] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the 8th annual ACM-SIAM symposium on Discrete algorithms (SODA)*, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[17] S. Bachthaler and D. Weiskopf, "Continuous scatterplots," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1428–1435, 2008.

[18] L. Backstrom, "Serving a billion personalized news feeds," in *The 12th International Workshop on Mining and Learning with Graphs at the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, ACM, 2016.

[19] E. Bakke and D. Karger, "Expressive query construction through direct manipulation of nested relational results," in *Proceedings of the 2016 International Conference on Management of Data (SIGMOD)*, ACM, 2016, pp. 1377–1392.

[20] E. Bakke, D. R. Karger, and R. C. Miller, "Automatic layout of structured hierarchical reports," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2586–2595, 2013.

[21] R. Barga, V. Fontama, and W. H. Tok, *Predictive analytics with Microsoft Azure machine learning (2nd Edition)*. Apress, 2015.

[22] S. Barocas and A. D. Selbst, "Big data's disparate impact," *California Law Review*, vol. 104, pp. 671–732, 2016.

[23] C. Batini, S. Ceri, and S. B. Navathe, *Conceptual Database Design: An Entity-Relationship Approach*. Benjamin Cummings, 1992.

[24] A. Beutel, J. Chen, T. Doshi, H. Qian, A. Woodruff, C. Luu, P. Kreitmann, J. Bischof, and E. H. Chi, "Putting fairness principles into practice: Challenges, metrics, and improvements," in *AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society (AIES)*, 2019.

[25] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword searching and browsing in databases using banks," in *Proceedings of the 18th International Conference on Data Engineering*, IEEE, 2002, pp. 431–440.

[26] D. Britz, "Implementing a CNN for text classification in TensorFlow," 2015. Accessed: June 26, 2017. [Online]. Available: `http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow`.

[27] M. Brooks, S. Amershi, B. Lee, S. M. Drucker, A. Kapoor, and P. Simard, "FeatureInsight: Visual support for error-driven feature ideation in text classification," in *IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE, 2015, pp. 105–112.

[28] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Conference on Fairness, Accountability and Transparency (FAT*)*, ACM, 2018, pp. 77–91.

[29] M. Buoncristiano, G. Mecca, E. Quintarelli, M. Roveri, D. Santoro, and L. Tanca, "Database challenges for exploratory computing," *ACM SIGMOD Record*, vol. 44, no. 2, pp. 17–22, 2015.

[30] Á. A. Cabrera, W. Epperson, F. Hohman, M. Kahng, J. Morgenstern, and D. H. Chau, "FairVis: Visual analytics for discovering intersectional bias in machine learning," in *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE, 2019.

[31] S. Carter and M. Nielsen, "Using artificial intelligence to augment human intelligence," *Distill*, 2017.

[32] T. Catarci, "What happened when database researchers met usability," *Information Systems*, vol. 25, no. 3, pp. 177–212, 2000.

[33] T. Catarci, M. F. Costabile, S. Levialdi, and C. Batini, "Visual query systems for databases: A survey," *Journal of Visual Languages and Computing*, vol. 8, no. 2, pp. 215–260, 1997.

[34] T. Catarci, G. Santucci, and M. Angelaccio, "Fundamental graphical primitives for visual query languages," *Information Systems*, vol. 18, no. 2, pp. 75–98, 1993.

[35] T. Catarci, G. Santucci, and J. Cardiff, "Graphical interaction with heterogeneous databases," *The VLDB journal*, vol. 6, no. 2, pp. 97–120, 1997.

[36] U. Cetintemel, M. Cherniack, J. DeBrabant, Y. Diao, K. Dimitriadou, A. Kalinin, O. Papaemmanouil, and S. B. Zdonik, "Query steering for interactive data exploration," in *Proceedings of the 6th Biennial Conference on Innovative Data Systems Research (CIDR)*, 2013.

[37] K. S.-P. Chang and B. A. Myers, "Using and exploring hierarchical data in spreadsheets," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM, 2016, pp. 2497–2507.

[38] B.-C. Chen, L. Chen, Y. Lin, and R. Ramakrishnan, "Prediction cubes," in *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, 2005, pp. 982–993.

[39] N.-C. Chen, J. Suh, J. Verwey, G. Ramos, S. Drucker, and P. Simard, "AnchorViz: Facilitating classifier error discovery through interactive semantic data exploration," in *Proceedings of the 23rd International Conference on Intelligent User Interfaces (IUI)*, ACM, 2018, pp. 269–280.

[40] P. P.-S. Chen, "The entity-relationship model: Toward a unified view of data," *ACM Transactions on Database Systems*, vol. 1, no. 1, pp. 9–36, 1976.

[41] Y. Chen, W. Wang, Z. Liu, and X. Lin, "Keyword search on structured and semi-structured data," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, ACM, 2009, pp. 1005–1010.

[42] R. H. Chiang, T. M. Barron, and V. C. Storey, "Reverse engineering of relational databases: Extraction of an eer model from a relational database," *Data & Knowledge Engineering*, vol. 12, no. 2, pp. 107–142, 1994.

[43] J. Choo, H. Lee, J. Kihm, and H. Park, "iVisClassifier: An interactive visual analytics system for classification based on supervised dimension reduction," in *IEEE Symposium on Visual Analytics Science and Technology (VAST)*, IEEE, 2010, pp. 27–34.

[44] A. Chouldechova, "Fair prediction with disparate impact: A study of bias in recidivism prediction instruments," *Big data*, vol. 5, no. 2, pp. 153–163, 2017.

[45] S. Chung, C. Park, S. Suh, K. Kang, J. Choo, and B. C. Kwon, "ReVACNN: Steering convolutional neural network via real-time visual analytics," in *Future of Interactive Learning Machines Workshop at the 30th Annual Conference on Neural Information Processing Systems (NIPS)*, 2016.

[46] Y. Chung, T. Kraska, N. Polyzotis, K. H. ; Tae, and S. E. Whang, "Slice finder: Automated data sclicing for model validation," *Proceedings of the 35th IEEE International Conference on Data Engineering (ICDE)*, pp. 1550–1553, 2019.

[47] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, ACM, 2016, pp. 191–198.

[48] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[49] M. Das, S. Amer-Yahia, G. Das, and C. Yu, "Mri: Meaningful interpretations of collaborative ratings," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, 2011.

[50] W. Dieterich, C. Mendoza, and T. Brennan, "COMPAS risk scales: Demonstrating accuracy equity and predictive parity," 2016.

[51] M. Dork, N. H. Riche, G. Ramos, and S. Dumais, "Pivotpaths: Strolling through faceted information spaces," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2709–2718, 2012.

[52] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[53] W. Dou, D. H. Jeong, F. Stukes, W. Ribarsky, H. R. Lipford, and R. Chang, "Recovering reasoning processes from user interactions," *IEEE Computer Graphics and Applications*, vol. 29, no. 3, pp. 52–61, 2009.

[54] J. J. Dudley and P. O. Kristensson, "A review of user interface design for interactive machine learning," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 8, no. 2, p. 8, 2018.

[55] J. Dunn, "Introducing FBLearner Flow: Facebook's AI backbone," *Facebook Code Blog*, 2016. Accessed: June 26, 2017. [Online]. Available: `https://code.facebook.com/posts/1072626246134461/introducing-fblearner-flow-facebook-s-ai-backbone/`.

[56] C. Dunne, N. Henry Riche, B. Lee, R. Metoyer, and G. Robertson, "GraphTrail: Analyzing large multivariate, heterogeneous networks while supporting exploration history," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, ACM, 2012, pp. 1663–1672.

[57] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. S. Zemel, "Fairness through awareness," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ACM, 2012, pp. 214–226.

[58] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, AAAI Press, 1996, pp. 226–231.

[59] J. Fan, G. Li, and L. Zhou, "Interactive SQL query suggestion: Making databases user-friendly," in *Proceedings of the 27th IEEE International Conference on Data Engineering (ICDE)*, IEEE, 2011, pp. 351–362.

[60] S. Few, *Show me the numbers: Designing tables and graphs to enlighten*. Analytics Press Oakland, CA, 2004.

[61] S. A. Friedler, C. Scheidegger, and S. Venkatasubramanian, "On the (im) possibility of fairness," *arXiv preprint arXiv:1609.07236*, 2016.

[62] M. Gleicher, "Explainers: Expert explorations with crafted projections," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2042–2051, 2013.

[63] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts, "Visual comparison for information visualization," *Information Visualization*, vol. 10, no. 4, pp. 289–309, 2011.

[64] G. Goh, "Why momentum really works," *Distill*, 2017.

[65] H. Gonzalez, A. Y. Halevy, C. S. Jensen, A. Langen, J. Madhavan, R. Shapley, W. Shen, and J. Goldberg-Kidon, "Google fusion tables: Web-centered data manage-

ment and collaboration," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, ACM, 2010, pp. 1061–1066.

[66] I. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[67] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 2672–2680.

[68] D. Gotz and M. X. Zhou, "Characterizing users' visual analytic activity for insight provenance," *Information Visualization*, vol. 8, no. 1, pp. 42–55, 2009.

[69] P. J. Guo, "Online python tutor: Embeddable web-based program visualization for cs education," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, ACM, 2013, pp. 579–584.

[70] M. Gyssens, J. Paredaens, J. Van den Bussche, and D. V. Gucht, "A graph-oriented object database model," *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, no. 4, pp. 572–586, 1994.

[71] M. Hardt, E. Price, N. Srebro, *et al.*, "Equality of opportunity in supervised learning," in *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS)*, 2016, pp. 3315–3323.

[72] A. W. Harley, "An interactive node-link visualization of convolutional neural networks," in *Proceedings of the 11th International Symposium on Visual Computing*, 2015, pp. 867–877.

[73] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[74] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, and J. Q. Candela, "Practical lessons from predicting clicks on ads at Facebook," in *Proceedings of the 8th International Workshop on Data Mining for Online Advertising at the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, ACM, 2014, pp. 1–9.

[75] A. Hindupur, "The GAN Zoo: A list of all named GANs!" 2017. Accessed: Mar. 31, 2018. [Online]. Available: `https://deephunt.in/the-gan-zoo-79597dc8c347`.

[76] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau, "Visual analytics in deep learning: An interrogative survey for the next frontiers," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 8, pp. 2674–2693, 2019.

[77] K. Holstein, J. Wortman Vaughan, H. Daumé III, M. Dudik, and H. Wallach, "Improving fairness in machine learning systems: What do industry practitioners need?" In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ACM, 2019, 600:1–600:16.

[78] V. Hristidis and Y. Papakonstantinou, "Discover: Keyword search in relational databases," in *Proceedings of the 28th International Conference on Very Large Data Bases*, 2002, pp. 670–681.

[79] C. D. Hundhausen, S. A. Douglas, and J. T. Stasko, "A meta-study of algorithm visualization effectiveness," *Journal of Visual Languages & Computing*, vol. 13, no. 3, pp. 259–290, 2002.

[80] S. Idreos, O. Papaemmanouil, and S. Chaudhuri, "Overview of data exploration techniques," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ACM, 2015, pp. 277–281.

[81] H. V. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu, "Making database systems usable," in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, ACM, 2007, pp. 13–24.

[82] H. V. Jagadish, A. Nandi, and L. Qian, "Organic databases," in *Databases in Networked Information Systems*, Springer, 2011, pp. 49–63.

[83] M. Jayapandian and H. V. Jagadish, "Automated creation of a forms-based database query interface," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 695–709, 2008.

[84] M. Joglekar, H. Garcia-Molina, and A. Parameswaran, "Interactive data exploration with smart drill-down," in *Proceedings of the 32nd International Conference on Data Engineering (ICDE)*, 2016, pp. 906–917.

[85] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, 2017.

[86] M. Kahng, P. Andrews, A. Kalro, and D. H. Chau, "ActiVis: Visual exploration of industry-scale deep neural network models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 88–97, 2018.

[87] M. Kahng and D. H. Chau, "How does visualization help people learn deep learning? evaluation of GAN Lab," in *Workshop on Evaluation of Interactive Visual Machine Learning Systems (EVIVA-ML) at IEEE VIS*, 2019.

[88] M. Kahng, D. Fang, and D. H. Chau, "Visual exploration of machine learning results using data cube analysis," in *Proceedings of the Workshop on Human-In-the-Loop Data Analytics (HILDA) at the 2016 ACM International Conference on Management of Data (SIGMOD)*, ACM, 2016, 1:1–1:6.

[89] M. Kahng, S. B. Navathe, J. T. Stasko, and D. H. Chau, "Interactive browsing and navigation in relational databases," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 1017–1028, 2016.

[90] M. Kahng, N. Thorat, D. H. Chau, F. B. Viégas, and M. Wattenberg, "GAN Lab: Understanding complex deep generative models using interactive visual experimentation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 310–320, 2019.

[91] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi, "Distributed and interactive cube exploration," in *Proceedings of the 30th International Conference on Data Engineering (ICDE)*, IEEE, 2014, pp. 472–483.

[92] H. Kang, C. Plaisant, B. Lee, and B. B. Bederson, "Netlens: Iterative exploration of content-actor network data," *Information Visualization*, vol. 6, no. 1, pp. 18–31, 2007.

[93] A. Karpathy, "ConvNetJS MNIST demo," 2016. Accessed: Mar. 31, 2018. [Online]. Available: `https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html`.

[94] M. Kearns, S. Neel, A. Roth, and Z. S. Wu, "Preventing fairness gerrymandering: Auditing and learning for subgroup fairness," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018, pp. 2569–2577.

[95] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, 1746–1751.

[96] J. M. Kleinberg, S. Mullainathan, and M. Raghavan, "Inherent trade-offs in the fair determination of risk scores," in *Proceedings of the 8th Innovations in Theoretical Computer Science (ITCS)*, 2017.

[97] J. Krause, A. Dasgupta, J. Swartz, Y. Aphinyanaphongs, and E. Bertini, "A workflow for visual diagnostics of binary classifiers using instance-level explanations,"

in *IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE, 2017.

[98] J. Krause, A. Perer, and E. Bertini, "INFUSE: Interactive feature selection for predictive modeling of high dimensional data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1614–1623, 2014.

[99] ——, "A user study on the effect of aggregating explanations for interpreting machine learning models," in *Workshop on Interactive Data Exploration and Analytics (IDEA) at the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.

[100] J. Krause, A. Perer, and K. Ng, "Interacting with predictions: Visual inspection of black-box machine learning models," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM, 2016, pp. 5686–5697.

[101] J. Krause, A. Perer, and H. Stavropoulos, "Supporting iterative cohort construction with visual temporal queries," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 91–100, 2016.

[102] T. Kulesza, M. Burnett, W.-K. Wong, and S. Stumpf, "Principles of explanatory debugging to personalize interactive machine learning," in *Proceedings of the 20th International Conference on Intelligent User Interfaces (IUI)*, ACM, 2015, pp. 126–137.

[103] T. Kulesza, S. Stumpf, W.-K. Wong, M. M. Burnett, S. Perona, A. Ko, and I. Oberst, "Why-oriented end-user debugging of naive Bayes text classification," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 1, no. 1, 2:1–2:31, 2011.

[104] A. Kumar, R. McCann, J. Naughton, and J. M. Patel, "Model selection management systems: The next frontier of advanced analytics," *ACM SIGMOD Record*, 2015.

[105] A. Kumar, J. Naughton, and J. M. Patel, "Learning generalized linear models over normalized data," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, ACM, 2015, pp. 1969–1984.

[106] M. J. Kusner, J. Loftus, C. Russell, and R. Silva, "Counterfactual fairness," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, 2017, pp. 4066–4076.

[107] H. Lakkaraju, E. Kamar, R. Caruana, and E. Horvitz, "Identifying unknown unknowns in the open world: Representations and policies for guided exploration," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, AAAI Press, 2017, pp. 2124–2132.

[108] Y. LeCun, "Answer to "what are some recent and potentially upcoming breakthroughs in deep learning?"" *Quora*, 2016. Accessed: Mar. 31, 2018. [Online]. Available: https://www.quora.com/What-are-some-recent-and-potentially-upcoming-breakthroughs-in-deep-learning.

[109] F. Li and H. V. Jagadish, "Constructing an interactive natural language interface for relational databases," *Proceedings of the VLDB Endowment*, vol. 8, no. 1, pp. 73–84, 2014.

[110] F. Li, T. Pan, and H. V. Jagadish, "Schema-free SQL," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ACM, 2014, pp. 1051–1062.

[111] F.-F. Li and J. Li, "Cloud AutoML: Making AI accessible to every business," *Google Blog*, 2018. Accessed: Sept. 21, 2019. [Online]. Available: https://www.blog.google/topics/google-cloud/cloud-automl-making-ai-accessible-everybusiness.

[112] X. Li and D. Roth, "Learning question classifiers," in *Proceedings of the 19th International Conference on Computational Linguistics*, 2002, pp. 1–7.

[113] B. Y. Lim and A. K. Dey, "Toolkit to support intelligibility in context-aware applications," in *Proceedings of the 12th ACM International Conference on Ubiquitous Computing (UbiComp)*, ACM, 2010, pp. 13–22.

[114] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 145–151, 1991.

[115] Z. C. Lipton, "The mythos of model interpretability," in *Proceedings of the 2016 ICML Workshop on Human Interpretability in Machine Learning at the 33rd International Conference on Machine Learning (ICML)*, 2016.

[116] B. Liu and H. V. Jagadish, "A spreadsheet algebra for a direct data manipulation query interface," in *Proceedings of the IEEE 25th International Conference on Data Engineering (ICDE)*, IEEE, 2009, pp. 417–428.

[117] M. Liu, J. Shi, K. Cao, J. Zhu, and S. Liu, "Analyzing the training processes of deep generative models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 77–87, 2018.

[118] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu, "Towards better analysis of deep convolutional neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 91–100, 2017.

[119] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci, "Visualizing high-dimensional data: Advances in the past decade," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 3, pp. 1249–1268, 2017.

[120] Z. Liu, B. Jiang, and J. Heer, "Immens: Real-time visual querying of big data," *Computer Graphics Forum (Proceedings of EuroVis)*, vol. 32, no. 3, pp. 421–430, 2013.

[121] Z. Liu, S. B. Navathe, and J. T. Stasko, "Network-based visual analysis of tabular data," in *IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE, 2011, pp. 41–50.

[122] Z. Liu and J. Stasko, "Mental models, visual reasoning and interaction in information visualization: A top-down perspective," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 999–1008, 2010.

[123] Y. Lu, R. Garcia, B. Hansen, M. Gleicher, and R. Maciejewski, "The state-of-the-art in predictive visual analytics," *Computer Graphics Forum (Proceedings of EuroVis)*, vol. 36, no. 3, pp. 539–562, 2017.

[124] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[125] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, 2017, pp. 2813–2821.

[126] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. M. Hrafnkelsson, T. Boulos, and J. Kubica, "Ad click prediction: A view from the trenches," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, ACM, 2013, pp. 1222–1230.

[127] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," in *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[128] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu, "Understanding hidden memories of recurrent neural networks," in *IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2017.

[129] Y. Ming, H. Qu, and E. Bertini, "RuleMatrix: Visualizing and understanding classifiers with rules," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 342–352, 2019.

[130] A. Nandi and H. V. Jagadish, "Guided interaction: Rethinking the query-result paradigm," *Proceedings of the VLDB Endowment*, vol. 4, no. 12, pp. 1466–1469, 2011.

[131] A. Nandi, L. Jiang, and M. Mandel, "Gestural query specification," *Proceedings of the VLDB Endowment*, vol. 7, no. 4, pp. 289–300, 2013.

[132] A. Nandi, C. Yu, P. Bohannon, and R. Ramakrishnan, "Data cube materialization and mining over mapreduce," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 10, pp. 1747–1759, 2012.

[133] T. Naps, S. Cooper, B. Koldehofe, C. Leska, G. Rößling, W. Dann, A. Korhonen, L. Malmi, J. Rantakokko, R. J. Ross, J. Anderson, R. Fleischer, M. Kuittinen, and M. McNally, "Evaluating the educational impact of visualization," *ACM SIGCSE Bulletin*, vol. 35, no. 4, pp. 124–136, 2003.

[134] T. L. Naps, G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, and J. A. Velázquez-Iturbide, "Exploring the role of visualization and engagement in computer science education," *ACM SIGCSE Bulletin*, vol. 35, no. 2, pp. 131–152, 2003.

[135] C. North, "Toward measuring visualization insight," *IEEE Computer Graphics and Applications*, vol. 26, no. 3, pp. 6–9, 2006.

[136] B. Nushi, E. Kamar, E. Horvitz, and D. Kossmann, "On human intellect and machine failures: Troubleshooting integrative machine learning systems," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017, pp. 1017–1025.

[137] C. Olah, "Neural networks, manifolds, and topology," 2014. Accessed: Mar. 31, 2018. [Online]. Available: `http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/`.

[138] C. Olah and S. Carter, "Research debt," *Distill*, 2017.

[139] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev, "The building blocks of interpretability," *Distill*, 2018.

[140] L. Parsons, E. Haque, and H. Liu, "Subspace clustering for high dimensional data: A review," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 90–105, 2004.

[141] K. Patel, N. Bancroft, S. M. Drucker, J. Fogarty, A. J. Ko, and J. Landay, "Gestalt: Integrated support for implementation and analysis in machine learning," in *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology (UIST)*, ACM, 2010, pp. 37–46.

[142] K. Patel, J. Fogarty, J. A. Landay, and B. Harrison, "Investigating statistical machine learning as a tool for software development," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, ACM, 2008, pp. 667–676.

[143] N. Pezzotti, T. Höllt, J. Van Gemert, B. P. Lelieveldt, E. Eisemann, and A. Vilanova, "DeepEyes: Progressive visual analytics for designing deep neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 98–108, 2018.

[144] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data lifecycle challenges in production machine learning: A survey," *ACM SIGMOD Record*, vol. 47, no. 2, pp. 17–28, 2018.

[145] E. D. Ragan, A. Endert, J. Sanyal, and J. Chen, "Characterizing provenance in visualization and data analysis: An organizational framework of provenance types and purposes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 31–40, 2015.

[146] V. Ramalingam, D. LaBelle, and S. Wiedenbeck, "Self-efficacy and mental models in learning to program," *ACM SIGCSE Bulletin*, vol. 36, no. 3, 2004.

[147] P. E. Rauber, S. G. Fadel, A. X. Falcao, and A. C. Telea, "Visualizing the hidden activity of artificial neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 101–110, 2017.

[148] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams, "Squares: Supporting interactive performance analysis for multiclass classifiers," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 61–70, 2017.

[149] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, ACM, 2016, pp. 1135–1144.

[150] B. Rosenberg, "Machine learning crash course," *Google Developers Blog*, 2018. Accessed: Mar. 31, 2018. [Online]. Available: `https://developers.googleblog.com/2018/03/machine-learning-crash-course.html`, 2018.

[151] M. A. Roth, H. F. Korth, and A. Silberschatz, "Extended algebra and calculus for nested relational databases," *ACM Transactions on Database Systems*, vol. 13, no. 4, pp. 389–417, 1988.

[152] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS)*, 2016, pp. 2234–2242.

[153] P. Saraiya, C. A. Shaffer, D. S. McCrickard, and C. North, "Effective features of algorithm visualizations," in *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, ACM, 2004, pp. 382–386.

[154] S. Sarawagi and G. Sathe, "I3: Intelligent, interactive investigation of olap data cubes," *ACM SIGMOD Record*, vol. 29, no. 2, p. 589, 2000.

[155] H.-J. Schek and M. H. Scholl, "The relational model with relation-valued attributes," *Information systems*, vol. 11, no. 2, pp. 137–147, 1986.

[156] D. Schweitzer and W. Brown, "Interactive visualization for the active learning classroom," *ACM SIGCSE Bulletin*, vol. 39, no. 1, pp. 208–212, 2007.

[157] D Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," in *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)*, 2015, pp. 2494–2502.

[158] J. F. Sequeda, M. Arenas, and D. P. Miranker, "On directly mapping relational databases to rdf and owl," in *Proceedings of the 21st international conference on World Wide Web*, ACM, 2012, pp. 649–658.

[159] C. A. Shaffer, M. L. Cooper, A. J. D. Alon, M. Akbar, M. Stewart, S. Ponce, and S. H. Edwards, "Algorithm visualization: The state of the field," *ACM Transactions on Computing Education*, vol. 10, no. 3, p. 9, 2010.

[160] B Shneiderman, "Direct manipulation: A step beyond programming languages," *IEEE Computer*, vol. 16, pp. 57–69, 1983.

[161] M. Singh, M. J. Cafarella, and H. V. Jagadish, "Dbexplorer: Exploratory search in databases," in *Proceedings of the 19th International Conference on Extending Database Technology (EDBT)*, 2016, pp. 89–100.

[162] D. Smilkov, S. Carter, D Sculley, F. B. Viegas, and M. Wattenberg, "Direct-manipulation visualization of deep networks," in *Workshop on Visualization for Deep Learning at the 33rd International Conference on Machine Learning (ICML)*, 2016.

[163] D. Smilkov, N. Thorat, C. Nicholson, E. Reif, F. B. Viégas, and M. Wattenberg, "Embedding Projector: Interactive visualization and interpretation of embeddings," in *Workshop on Interpretable Machine Learning in Complex Systems at the 30th Annual Conference on Neural Information Processing Systems (NIPS)*, 2016.

[164] C. Stolte, D. Tang, and P. Hanrahan, "Polaris: A system for query, analysis, and visualization of multidimensional relational databases," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 52–65, 2002.

[165] ——, "Multiscale visualization using data cubes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 176–187, 2003.

[166] M. Stonebraker and D. Moore, *Object Relational DBMSs: The Next Great Wave*. Morgan Kaufmann Publishers Inc., 1995.

[167] H. Strobelt, S. Gehrmann, M. Behrisch, A. Perer, H. Pfister, and A. M. Rush, "Seq2Seq-Vis: A visual debugging tool for sequence-to-sequence models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 353–363, 2019.

[168] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush, "LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 667–676, 2018.

[169] Y. Sun and J. Han, "Mining heterogeneous information networks: Principles and methodologies," *Synthesis Lectures on Data Mining and Knowledge Discovery*, vol. 3, no. 2, pp. 1–159, 2012.

[170] J. Talbot, B. Lee, A. Kapoor, and D. S. Tan, "EnsembleMatrix: Interactive visualization to support machine learning with multiple classifiers," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, ACM, 2009, pp. 1283–1292.

[171] L. Theis, A. v. d. Oord, and M. Bethge, "A note on the evaluation of generative models," in *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, 2016.

[172] J. Tyszkiewicz, "Spreadsheet as a relational database engine," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, ACM, 2010, pp. 195–206.

[173] S. Van Den Elzen and J. J. Van Wijk, "BaobabView: Interactive construction and analysis of decision trees," in *IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE, 2011, pp. 151–160.

[174] M. Veale, M. Van Kleek, and R. Binns, "Fairness and accountability design needs for algorithmic support in high-stakes public sector decision-making," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ACM, 2018, 440:1–440:14.

[175] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2015, pp. 3156–3164.

[176] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the gdpr," *Harvard Journal of Law & Technology*, vol. 31, no. 2, pp. 841–887, 2017.

[177] J. Wang, L. Gou, H. Yang, and H.-W. Shen, "GANViz: A visual analytics approach to understand the adversarial game," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 6, pp. 1905–1917, 2018.

[178] M. Wattenberg, F. Vigas, and I. Johnson, "How to use t-SNE effectively," *Distill*, 2016.

[179] D. S. Weld and G. Bansal, "The challenge of crafting intelligible intelligence," *Communications of the ACM*, vol. 62, no. 6, pp. 70–79, 2019.

[180] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Vigas, and J. Wilson, "The What-If Tool: Interactive probing of machine learning models," *IEEE Transactions on Visualization and Computer Graphics (Early Access)*, 2019.

[181] J. Wexler, "Facets: An open source visualization tool for machine learning training data," *Google AI Blog*, 2017. Accessed: Sept. 21, 2019. [Online]. Available: `https://ai.googleblog.com/2017/07/facets-open-source-visualization-tool.html`.

[182] B. Wilson, J. Hoffman, and J. Morgenstern, "Predictive inequity in object detection," *arXiv preprint arXiv:1902.11097*, 2019.

[183] K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Mané, D. Fritz, D. Krishnan, F. B. Viégas, and M. Wattenberg, "Visualizing dataflow graphs of deep learning models in TensorFlow," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 1–12, 2018.

[184] K.-W. Wu, C.-S. Ferng, C.-H. Ho, A.-C. Liang, C.-H. Huang, W.-Y. Shen, J.-Y. Jiang, M.-H. Yang, T.-W. Lin, C.-P. Lee, P.-H. Kung, C.-E. Wang, T.-W. Ku, C.-Y. Ho, Y.-S. Tai, I.-K. Chen, W.-L. Huang, C.-P. Chou, T.-J. Lin, H.-J. Yang, Y.-K. Wang, C.-T. Li, S.-D. Lin, and H.-T. Lin, "A two-stage ensemble of diverse models for advertisement ranking in kdd cup 2012," in *KDD Cup Workshop at the 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2012.

[185] T. Wu, M. T. Ribeiro, J. Heer, and D. S. Weld, "Errudite: Scalable, reproducible, and testable error analysis," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019, pp. 747–763.

[186] D. Xu, S. Yuan, L. Zhang, and X. Wu, "Fairgan: Fairness-aware generative adversarial networks," in *Proceedings of the 2018 IEEE International Conference on Big Data*, IEEE, 2018, pp. 570–575.

[187] Q. Yang, J. Suh, N.-C. Chen, and G. Ramos, "Grounding interactive machine learning tool design in how non-experts actually build models," in *Proceedings of the 2018 Designing Interactive Systems Conference*, ACM, 2018, pp. 573–584.

[188] X. Yang, C. M. Procopiuc, and D. Srivastava, "Summarizing relational databases," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 634–645, 2009.

[189] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," in *Deep Learning Workshop at the 31st International Conference on Machine Learning (ICML)*, 2015.

[190] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, *et al.*, "Apache spark: A unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[191] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork, "Learning fair representations," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, PMLR, 2013.

[192] C. Zhang, A. Kumar, and C. Ré, "Materialization optimizations for feature selection workloads," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ACM, 2014, pp. 265–276.

[193] M. M. Zloof, "Query-by-example: A data base language," *IBM Systems Journal*, vol. 16, no. 4, pp. 324–343, 1977.